2019

Stageverslag Michiel Maas

AUTOMATISERING RAPPORTAGES

MICHIEL MAAS

OIS SOFTWAREMAKERS | Ampèrelaan 4, 2289 CD Rijswijk

Voorwoord:

Dit verslag gaat over mijn eerste echt werkervaring bij een software bedrijf. Het is een beschrijving van de stappen en keuzes die ik heb gemaakt in mijn proces, en bij motivatie daarbij. Ik heb veel geleerd over programmeren, ontwerpen, presenteren, keuzes maken, time management en kantoor humor. Dit was een van de meest leerzame momenten van mijn gehele opleiding. Bedankt OIS voor deze mooi kans!

Michiel Maas - Rijswijk - 5 juli 2019

Inhoudsopgaven

1 Inleiding

Als deel van de opleiding HBO-ICT van de Haagse Hogeschool volg ik een werkstage. Deze werkstage geeft de student werkervaring en bereid hem of haar voor op het leven na de studie. Mijn stage liep van 1 februari tot 5 juli, en vervulde ik bij software bedrijf OIS. Vijf dagen per week, acht uur per dag.

Mijn stage opdracht was het 'Automatiseren van Rapportages' bij het bedrijf OIS. En bedrijf klein in medewerkers, maar groot op de markt. Ik vond het leuk dat ik bij een bedrijf, dat zo veel klanten heeft, stage mocht lopen en een product mocht maken. De stage was erg leerzaam. Zowel op programeergebied als op sociaalgebied. Het werkleven is heel anders dan het studenten leven, en heeft mij veel geleerd over mijn toekomst.

In dit verslag beschrijf ik de stappen die ik heb genomen bij het ontwerpen van mijn applicatie. Ik beschrijf het onderzoek dat ik heb gedaan naar de verschillende tools die ik kon gebruiken, het ontwerpen van de hoe de software samen werkt en alle grote beslissingen die ik heb gemaakt.

2 Context

De stage heeft plaats gevonden bij ‘OIS Softwaremakers’, een programeerbedrijf in Rijswijk. OIS maakt een platform voor het verwerken van administratieve zaken, en heeft daarbij verschillende toepassingen ontwikkeld:

- SDS Medical – Een Medoc platform voor het registeren van ziekenhuis informatie
- ODW – Handelt Sociale Zaken bij verschillende gemeenten in Nederland
- ECManage – De marktleider op het gebied van het bedrijfskleding verkoop

Deze toepassingen worden verkocht aan klanten als abonnement, niet als een pakket zoals vaak gebruikelijk is. De waarde die OIS dan ook verkoopt komt uit de service. OIS bestaat daarom niet alleen uit ontwikkelaars, maar ook uit service medewerkers, technische beheerders, consultants en sales managers. In totaal werken er ongeveer 25 mensen bij OIS. Dit getal is klein voor wat het bedrijf allemaal neer zet. De directie van OIS bestaat uit drie mensen en is een team leider van de ontwikkelaars. De structuur is echter niet erg hiërarchisch

De werkhouding is nuchter, en valt niet voor de moderne hypes. Software wordt ontwikkeld voor de toekomst en op brede functionaliteit. De sfeer op kantoor in informeel. Zolang je je werk afkrijgt, en het werk goed gemaakt wordt, maakt het niet heel veel uit hoe laat je binnenkomt en of weg gaat. Lunch wordt gratis verzorgd en er wordt vaak gewandeld in de pauze.

3 Probleem Omschrijving

Periodiek worden er rapportages verstuurd naar de klanten van OIS. Deze rapportages hebben gegevens over het gebruik van de applicaties en beschrijft de voortgang van het ontwikkelingen van nieuwe features en het oplossen van bugs in de applicatie. Op dit moment worden deze rapportages gemaakt door de service medewerkers aan de hand van een aantal templates.

Het maken van de rapportages kost veel tijd. De bestanden met de gegevens moeten op de goede manier worden ingelezen. De rapportages zelf worden gemaakt aan de hand van wat scriptjes. Deze rapportages worden dan verstuurd over de mail. Deze stappen zijn erg foutgevoelig en gaan daarom ook regelmatig fout. Windows updates en foute invoer van gegevens zijn vaak de boosdoeners.

Dit proces, het generen en versturen van de rapportages, moet geautomatiseerd worden.

4 Huidige Situatie omschrijving:

4.1 Rapportages

Er zijn verschillende rapportages die periodiek verstuurd worden binnen OIS. Deze rapportages houden de informatie over de stand van zaken van de gebruikers van de applicaties. De rapportages hebben verschillende informatie over verschillende onderwerpen, toegepast op de applicatie.

4.1.1 ECManage Maand Rapportages

Iedere maand wordt er naar de ECManage klanten een overzicht gestuurd met het aantal Dragers en Orders van die, en de afgelopen 12 maanden.

Dit proces gebeurd aan de hand van een aantal macro's en .xml bestanden. Het is erg foutgevoelig en ligt er regelmatig uit door een vereiste Windows Update.

4.1.2 IRIS Rapportages

Iedere week wordt er naar alle OIS klanten een overzicht gestuurd met informatie over de bugfixes en wijzing verzoeken van de klant. Deze gegeven zijn aangesloten op de algemene meldingen database van OIS.

Dit proces gebeurd, net als de ECManage rapportage, aan de hand van een aantal .VBA scriptjes die macro's uitoefenen. Het hele proces duurt ongeveer 15 minuten, maar kan uitlopen tot een half uur.

4.1.3 Data Aggregatie

Veel data, zoals bijvoorbeeld de directe verkoopgegevens van ECManage, worden niet centraal opgeslagen in een database. Dit maakt het moeilijk om en goed globaal overzicht te krijgen van de stand van zaken.

5 Analyse

Het bedrijf is veel gegroeid de afgelopen jaren. Er zijn veel nieuwe klanten bijgekomen, wat er voor zorgt dat er meer rapportages gemaakt moeten worden. Het rapportage systeem is verouderd en is inefficiënt voor de huidige schaal. Er zijn af en toe patch oplossingen geweest, maar het was tijd dat er een nieuwe en goede oplossing kwam.

5.1 Rapportages

De huidige twee rapportages zijn die van de IRIS Meldingen en de ECManage maandelijkse overzichten.

5.1.1 ECManage Maandelijkse Rapportages

Een keer per maand wordt er een overzicht gestuurd naar alle klanten met hoeveel zij gebruik maken van de ECManage service. Ze krijgen een overzicht van alle omgevingen die zijn aangesloten aan hun systeem en hoeveel Dragers (mensen aangesloten op het systeem) en Orders (bestellen via het systeem) die hebben afgelopen maand. Er wordt een grafiek en een tabel gemaakt met deze cijfers van de afgelopen twaalf maanden.
Deze rapportages worden gemaakt aan de hand van een .XML bestand die iedere maand vanuit de Oracle database wordt gestuurd naar de medewerker die de overzichten genereerd. De gegevens van de .XML bestanden worden aan de hand van een aantal Macro's ingelezen in Excel bestanden en de gegevens worden verwerkt en berekend. Deze gegevens worden dan in een Word Template gezet samen met de grafieken gemaakt in de Excel bestanden. Het Word document wordt opgeslagen als .pdf en een concept email gezet. Elke email moet vervolgens met de hand worden verstuurd.

De huidige manier van werken is onhandig om een aantal redenen:

- Alle overzichten van de gegevens worden bewaard in Excel bestanden. Dit maakt het moeilijk om een goed overzicht te krijgen van de stand van zaken.
- In elke Excel tabel worden de gegevens van de afgelopen twaalf maanden bewaard, en van iedere maand worden alle bestanden bewaard. Dus een bepaald maand gegeven staat totaal in twaalf verschillende Excel bestanden. Die oudere bestanden zijn helemaal niet nodig, zeker niet met al die dubbele gegevens.
- In een ander Excel sheet wordt met de hand ontvangers van de e-mails toevoegt. Dit is erg fout gevoelig en kan voor fouten zorgen in het verstuurd proces.
- Sommige gegevens uit de Excel tabellen moeten met de hand overzet worden naar een ander overzichtsbestand die alle betaling bij houdt. Van dit bestand wordt maar een aantal keer per jaar een back-up gemaakt.
- Het systeem loopt vaak vast door Windows Updates die dan niet meer samen werken met de Macro's uit de bestanden.
- En de belangrijkste : hoe lang het duurt. Er moeten drie grote stappen ondernemen worden en tijdens deze stappen kan je geen gebruik van de computer. Als het fout gaat moet alles weer opnieuw vanaf begin af aan, en de e-mails moeten uiteindelijk allemaal met de hand verstuurd worden door op verzenden te klikken. Het complete proces kan soms wel 20 minuten duren, en dat is als alles goed gaat.

De macro's die gebruikt worden voor het maken van de rapportages zijn ook geanalyseerd. Het leek alsof de implementatie was gemaakt door iemand die haast had. Veel stappen waren gekopieerd van vorige stukjes code en er werden veel overbodige berekeningen en bewerkingen gedaan. Hier en daar stonden wat stukje die bedoelt waren als een Quick fix, dat stond zo beschrijven in de comments, maar die zijn in de afgelopen 4 jaar van gebruik niet verbeterd.

Bij elkaar duurt het proces ongeveer 45 minuten, maar uitlopen naar een uur of misschien zelf een halve dag wanneer een Windows Update nodig is.
Dit proces had dringend een update nodig.

5.1.2 IRIS Rapportages

Een keer per week worden er IRIS Rapportages gemaakt. In deze rapportages staan de werkzaamheden van afgelopen week beschrijven. Er staat beschrijven welke features zijn geïmplementeerd, welke bug reports zijn binnen gekomen en aan welke projecten er is gewerkt.

De rapportages worden gemaakt aan de hand van een Word macro. Bij het openen van het bestand moet de gebruik in een invul veld invullen welke week het is van het jaar, en welk jaar het is. Deze gegevens worden gebruikt om de correcte gegevens uit de database op te vragen. Deze gegevens worden vervolgens in een Word Template gezet en opslagen. Daarna wordt met een batch file alle rapportages verzonden naar de klanten. Dit wordt gedaan voor elk van de drie pakketten die OIS aanbied.

Dit proces is sneller dan het maken van de ECManage Rapportages, en gaat ook minder vaak fout. In totaal duurt het hele proces ongeveer 10 tot 15 minuten. Er vielen een paar dingen op tijdens de demonstratie:

- Het invoeren van het weeknummer en jaar is erg fout gevoelig. Als je een fout maakt moet die rapportage opnieuw gemaakt worden.
- Het versturen naar de klanten gaat met een Batch file. Dit is sneller dan bij de andere rapportage maar zorgt voor minder controle. Als je per ongelijk een fout heb gemaakt krijgt iedereen de verkeerde mail, je kan het proces niet gemakkelijk halverwege stoppen. Ook moet het bestand goed bijgehouden worden.

Over het algemeen oogde dit proces beter dan de ECManage Rapportage.

Tijdens het analyseren van de macro's die gebruikt worden bij het generen van de IRIS Rapportages was er een bug gevonden. Twee van de velden werden nooit ingevuld omdat er een varchar wordt gezocht in een veld waar een Integer moest. Dit zorgde er voor dat deze velden nooit werden ingevuld. Voor de afgelopen drie jaar zijn deze velden nooit ingevuld, en het is geen enkele klant opgevallen. Deze bug was gemakkelijk te fixen zodat het nog gebruikt kon worden. Dit was wel een teken van de urgentie van deze opdracht.

5.2 Tijd Besparend

Zoals hier boven uitgelegd is het proces erg tijd rovend. Het nieuwe proces moet zo min mogelijk onderhoud kosten en volledig autonoom kunnen werken. De applicatie moet zelf kijken wanneer het tijd is om gegevens op te slaan, rapportages te maken/versturen en overzichten te generen. Een beheerder zou deze instellingen, misschien klant specifiek, kunnen aanpassen.

5.3 Generieke Database

OIS ziet graag dat de uiteindelijke applicatie nog lang mee gaat, en dat ze het kunnen aansluiten op eventuele nieuwe systemen. Als iedere keer dat een nieuwe bron toegevoegd moet worden de database uitgebreid of aangepast moet worden is dat onhandig. Het is ingewikkeld, kost veel werk en in sommige gevallen bijna onmogelijk. De uiteindelijke applicatie moet op een generieke database draaien die gegevens van alle verschillende vormen en maten, die je van te voren niet kan weten, moet kunnen opslaan en daarna weer uithalen. Dit is belangrijk voor de functionaliteit en de levensduur van de applicatie.

5.4 Modulair

De uiteindelijke applicatie moet erg modulair zijn zodat het verschillende databronnen kan aan nemen en uitvoeren. Hiermee wordt bedoeld dat er gemakkelijk nieuwe features aan de applicatie kunnen worden toegevoegd. Wanneer er een nieuwe bron komt voor de data, moet deze gemakkelijk aangesloten worden aan het systeem. Wanneer er een andere uitdraai moet komen van de gegevens, moet dit ook met een losse module uitgevoerd kunnen worden en met weinig werk geïmplementeerd worden.

5.5 Rapportage Generatie

Veel van de rapportages die verstuurd worden op dit moment, worden verstuurd als PDF. Op dit moment gaat dat via Word, maar de bedoeling is dat deze stap overgeslagen wordt zodat het minder tijd gaat kosten voor de uiteindelijke applicatie. De rapportages hebben vaak ook afbeeldingen van grafieken en tabellen met gegevens. De rapportages mogen ook een nieuwe look krijgen, de huidige rapportages zijn verouderd in ontwerp.

5.6 Documentatie

Voor het geval dat de applicatie niet op tijd af komt is het ook belangrijk om ook uitgebreide documentatie van de applicatie. Door de grote omvang van het project is er een kans dat iemand anders het werk moet afmaken of uitbreiden.

5.7 Robuust

De uiteindelijke applicatie moet robuust zijn. Hier wordt bedoeld dat het lang bruikbaar moet zijn voor OIS. De applicatie moet makkelijk aanpasbaar zijn naar de wensen van OIS. Dit kan door het toevoegen van nieuwe functies en opties.

6 Requirements

Aan de hand van de analyse worden de requirements opgesteld voor de applicatie. De requirements zijn beschreven volgens de MoSCoW (Must Have, Should Have, Could Have, Would Have) methode.

- Must Have
  - Generen van verschillende soorten rapportages
  - Zelfstandig werkend proces
  - Goede documentatie voor zodat het project kan worden overgenomen
  - Robuust systeem, lang bruikbaar
  - Generieke database voor veel verschillende soorten data
- Should have
  - Beheerders dashboard voor specifieke aanpassingen
  - Modulaire onderdelen voor het toevoegen van nieuwe toepassingen
  - De applicatie moet goeden samenwerken met bestaande OIS implementaties
- Could Have
  - Meerdere mogelijke in- en output
  - Formaat onafhankelijke rapportages voor nieuwe bronnen
- Would Have
  - Klant specifieke instellingen

7 Ontdekkingsfase

In deze fase is er gekeken naar de beste oplossing voor de verschillende requirements. Voordat de ontwikkel fase kon beginnen moesten de pakket keuze gemaakt worden van de applicatie.

Het pakket bestaat uit:

- Ontwerp van bestand
- Ontwikkel taal
- Database Systeem

Na de pakket keuze is er een vertical prototype gemaakt om te testen of de pakket keuze correct was.

7.1 Opdelen opdracht

De applicatie was op te delen in 4 verschillende lagen. Elke laag is zou zo modulair mogelijk gemaakt worden, zodat de applicatie met verschillende input, verschillende output en verschillende manieren gemakkelijk gebruikt kan worden.

7.1.1 Laag 1: Data Collectie

De data die in de database moet staan komt van verschillende plekken en in verschillende formaten. De input stromen moeten ontvangen worden en getransleerd worden zodat ze uniform de database binnen kunnen komen. Omdat het aantal input stromen flexibel moet zijn is het belangrijk dat die volgens een makkelijke procedure gaat en dat er modulen toegevoegd en weg gehaald kunnen worden.

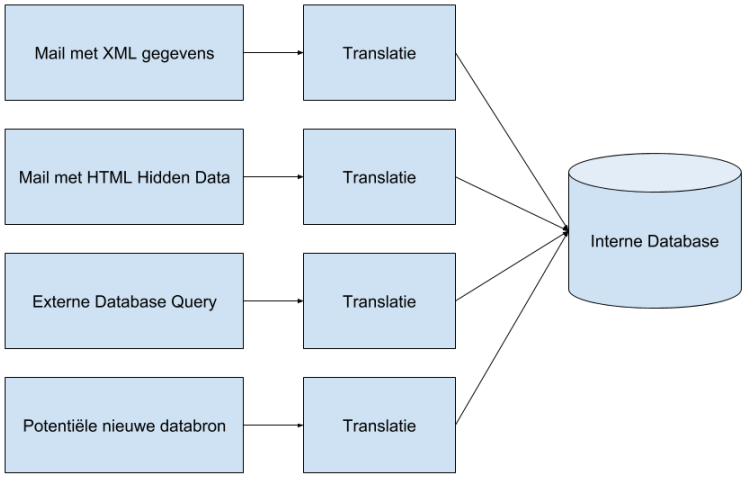


Figure 1 : Datacollectie-laag

7.1.2 Laag 2: Data Aggregatie

Van de interne database kunnen er verschillende overzichten gemaakt worden. De gegevens moeten nu eenmaal ook gebruikt worden. Met verschillende soorten query's en logic kunnen we verschillende overzichten generen met verschillende gegevens voor elke dat dat nodig is.

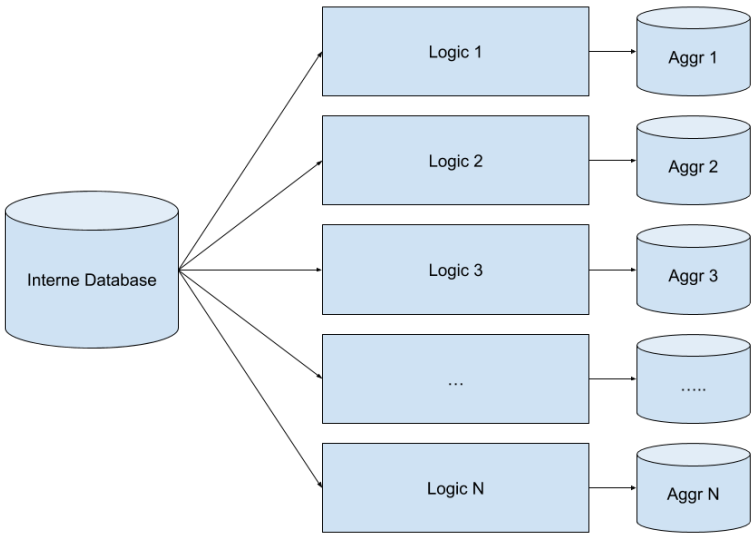


Figure 2 Data aggregatie-laag

7.1.3 Laag 3: Data Presentatie

De gemaakte aggregaties kunnen op verschillende manieren gepresenteerd worden voor hun verschillende doeleinden. Sommige moeten in PDF vorm verstuurd worden, andere zijn beter als .docx of html bestand

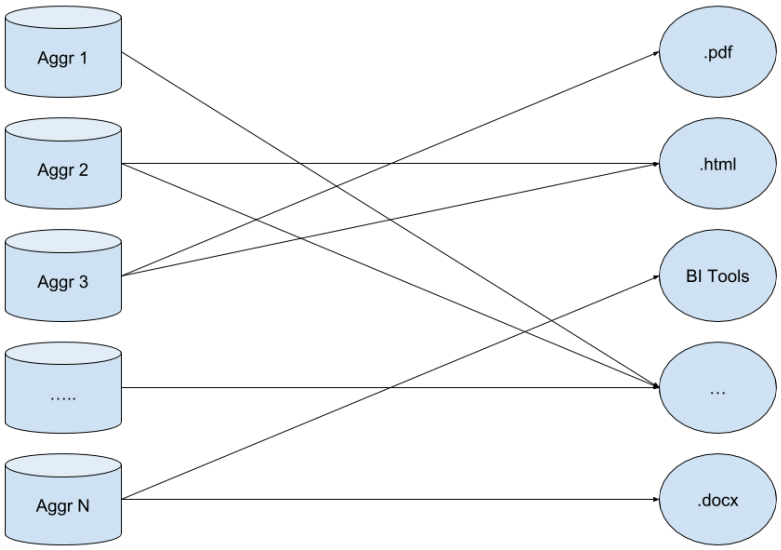


Figure 3 Datapresentie-laag

7.1.4 Laag 4: Data Distributie

De gemaakt overzichten moeten ook naar de goede klant en op de goede manier bestuurd worden. Deze laag zorgt er voor dat de gemaakte overzicht op het goede moment in de goede handen valt. Dit kan een keer per week via de mail, of een keer per maand uit geprint op de lokale printer.

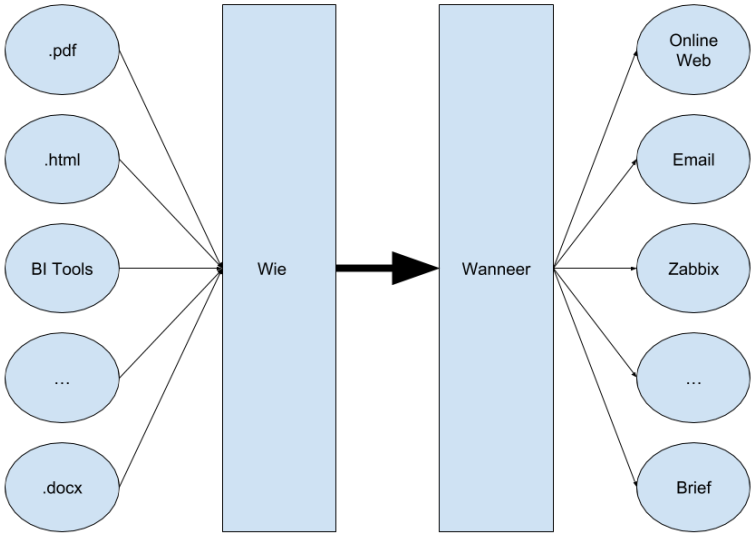


Figure 4 Datadistributie-laag

7.1.5 Modulair

Deze lagen moeten zo modulair mogelijk, zodat er verschillende combinaties gemaakt kunnen worden voor verschillende toepassingen wanneer nodig. Het aansluiten van een nieuwe databron, of het extraheren van een ander overzicht van gegevens zou zonder te veel moeite moeten lukken.

7.2 Database Systeem Keuze

De verwachting was dat de database in een Relationele Database gemaakt zou worden. De invoer moest erg generiek zijn, dus is het database model onafhankelijk gemaakt van invoer en enorm genormaliseerd. Eerst is een Database Ontwerp gemaakt (zie Bijlagen A), maar na een korte test run bleek bij deze dat het erg ingewikkeld was om de data correct in de database in te voeren en de correcte gegevens eruit te halen. Dit zorgde voor een zoektocht naar een alternatief.

Het probleem, de ingewikkeldheid van de RDBM, samen met de alternatieven werd aan de begeleider voor gelegd. Samen is besloten 3 proof-of-concepts te maken van 3 verschillende data modellen en vervolgens te kijken welke de beste keus is. De gekozen datamodellen waren RDB, ERA (Key-Value) en noSQL. Er is gekozen voor deze drie omdat ze veel van elkaar verschillen.

7.2.1 Relationele Database

RDB, de meest gebruikte vorm van datamodellen, en de verwachte keus. Omdat invoer volgens de opdracht zo generiek mogelijk moet is het model enorm genormaliseerd. Bij het modeleren van de database bleek dit een probleem. Toch is het de moeite waard op te onderzoeken of dit de beste keus is. Als Database Server is gekozen voor MariaDB. Dit was door ervaring met MariaDB en op aanbeveling van de stagebegeleider.

7.2.2 noSQL

'the "new" kid on the Block'. In plaats van data je data gestructureerd en verspreid opslaat, doe je het allemaal op een plek. Wanneer je snel jouw data in een database wil hebben, zonder dat je structuur je iets uitmaakt, is noSQL een goede keus. Wel is het zo dat, omdat je data bij elkaar staat en JOINs vaak niet ondersteund worden, er veel redundante data is.
JOINs zijn belangrijk om te doen, omdat er veel overzichten gemaakt moeten worden voor de Data Aggregatie vraag van de opdrachtgever. Als JOINen moeilijk is, kan het ook ingewikkeld zijn om de overzichten te genereren.
Er zijn twee vormen van noSQL gevonden: MongoDB en ArangoDB. MongoDB is de grootste in de noSQL sfeer en professioneel ingericht, en ArangoDB is een stuk kleiner maar had een paar mooie features. Beide zijn hieronder vergeleken.

7.2.2.1 MongoDB

De grootste speler in de noSQL wereld. Er zijn veel tutorial te vinden over MongoDB en ook heeft Stack Overflow veel documentatie.
Wel viel het me op dat de Database alleen met programeer code te benaderen was, en vaak de query's redelijk omslachtig leken.
Ook waren JOINs uit den boze. Om twee 'Collections' (Tables in SQL termen) met elkaar te JOINen moet je alle records uit de Collection vergelijken, wat zorgt veel werkgeheugen.
Het leek veel belovend door het grote aantal gebruikers, maar de functies stelde me een beetje teleur.

7.2.2.2 ArangoDB

ArangoDB is een schemaless, multimodel noSQL Database Model. Je kan niet alleen 'Documents' opslaan in de database, maar ook 'Graphs' die twee 'Documents' aan elkaar kunnen linken. Deze structuur ondersteund dus JOINs. Een groot pluspunt.
Verder heeft ArangoDB ook een eigen querying language (AQL) die de hele database kan doorzoeken. Deze querying language was aantrekkelijk omdat het meer functionaliteit leek te hebben dan die van MongoDB.
Daarboven op had ArangoDB ook een eigen Micro Service Integration (Foxx). Met Foxx kan je

gemakkelijk API Calls schrijven voor op de database. Op deze manier kan je je vragen naar de database structuren en zelfs schema's afdwingen in de gegevens die je opslaat.

Doordat ArangoDB een eigen querying language heeft met veel functionaliteit, JOINs van data goed ondersteund worden en dat het een eigen Micro Service Integration heeft, viel de keuze op ArangoDB. Het is dan wel minder bekend en minder over te vinden, maar de documentation was uitgebreid en de functionaliteit woog niet op tegen de informatie over MongoDB.

7.2.3 ERA (Key Value)

Ik wilde graag nog een derde Data Model onderzoeken of deze makkelijke te gebruiken was. De noSQL en de RDBMS oplossingen waren voldoende, maar er is gekeken naar een derde voor de zekerheid. De begeleider liet vallen dat een Key-Value systeem misschien kon werken, dus is hier onderzoek naar gedaan.

Het onderzoek gaf een paper over de implementatie van een Key-Value database in een T-SQL (Lekberg & Danielsson, 2013). Dit kwam exact overeen met wat gezocht werd.

Het beschreven probleem: een generieke database een noSQL implementatie in SQL.

7.2.4 Eisen aan Proof-of-concept

Om alles goed te testen heb zijn een aantal eisen opgesteld aan het proof-of-concept om een goed besluit te maken.

- Gebruik de zelfde data set
    - Er is een proef dataset gemaakt die door alle drie de verschillende database modellen gebruikt kan worden om te kijken of alles (gemakkelijk) opgeslagen kan worden. En dat we ook de zelfde resultaten kunnen krijgen. Ook zijn er in de dataset een aantal uitzonderingen opgenomen om de dataset op te kijken hoe er mee om wordt gegaan. (Zie Bijlagen B).
- Maak de zelfde drie overzichten:
    - Maak een overzicht van de IRIS Meldingen
    - Maak een overzicht van Sales Totals, de gegevens van afgelopen jaar over alle leveranciers.
    - Maak een overzicht voor elke leverancier van afgelopen jaar.

Bij het controleren wordt er gekeken naar een aantal criteria. Deze zijn belangrijk voor de keuze voor het uit eindelijke project , maar ook voor OIS, de opdrachtgever. De hoofd punten zijn onderverdeeld in kleinere deelonderwerpen.

- Gemak van gebruik
    - Hoe gemakkelijk is het op nieuwe data in te voeren?
    - Hoe gemakkelijk is het om goede data uit te halen
    - Hoe makkelijk is het op data te updaten?
    - Hoe veel verschillende soorten data kan erin?
- CPU
    - Hoe lang duren de query's?
    - Hoe veel CPU / werk vermogen kost het?
    - Hoe goed kunnen de query's nog worden verbeterd?
- Aansluiting in het bedrijf
    - Hoe goed werken de andere database / gegeven invoer met deze vorm?
    - Hoe makkelijk is het voor een volgende gebruiker om door te gaan met deze vorm.

Met deze eisen zijn de drie proof-of-concepts gemaakt.

7.2.5 RDBM:

Het modeleren van een generieke database in SQL is redelijk ingewikkeld. Je kan van te voren niet verwachten welke vorm de data zal aannemen, dus moet je alle verschillende toe staan. Dit proces is stap voor stap aangepakt.
De uitgangspunten waren de bronnen die nu gebruikt worden voor de rapportages. Waar een overeenkomsten was te vinden dus de rapportages, die ook kan worden verwachten in potentiele nieuwe rapportages, kon deze gebruikt worden. Ook sommige entiteiten hoefde niet gegeneraliseerd te worden, zoals bedrijf en persoon.
Daarna is ook gebruikt gemaakt van literatuur om te leren over het maken van een goede (generieke) database (West & Julian, 1996) . Aan de hand van de literatuur is het database ontwerp nog aangepast. Nadat een voldoende database was ontworpen, werd deze overgezet in HeidiSQL, een RDBMS, en ingeladen met de dataset. Daarna moesten de correcte gegevens weer uit de dataset gehaald worden.

Met dit ontwerp bleek het een hele grote klus. Omdat alles zo generiek en genormaliseerd is moesten veel Tables aanroepen worden, JOINs gemaakt en Views gebruiken worden. De lastige schakel in het ontwerp was de opslag van attributen bij objecten.

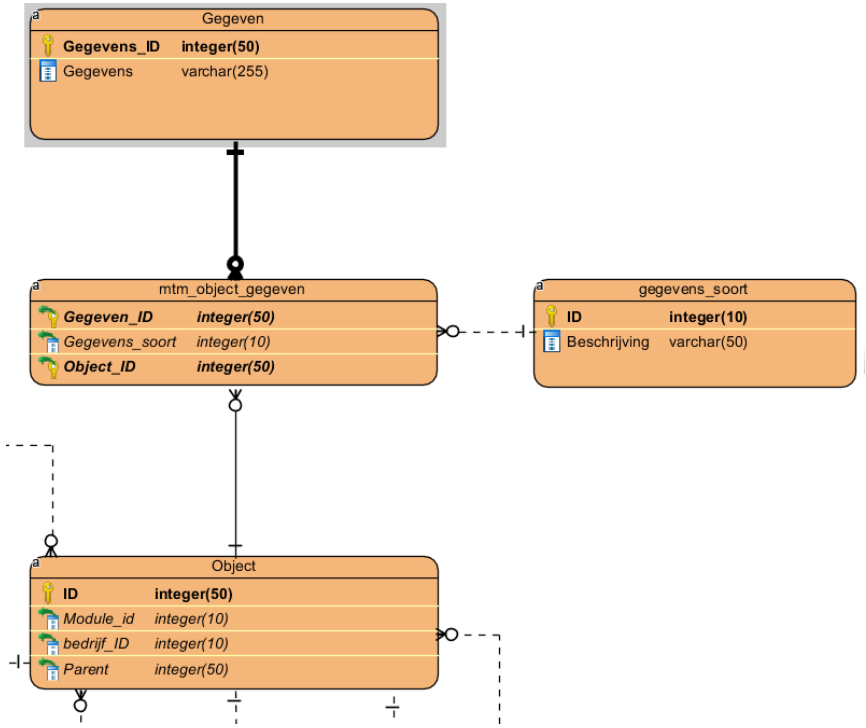


Figure 5: RDB Datamodel Object naar Gegevens

Omdat van te voren niet te weten is hoeveel attributen een record heeft, is een Many-to-Many connectie gemaakt tussen ‘Object’ (Het hoofd record waar alles aan wordt gelinkt) en ‘Gegeven’ (Een attribuut van het object). De Many-to-Many tabel heeft nog een extra kolom, Gegevens_soort, die aangeeft wat voor soort attribuut het is waar die is gelinkt. Als je twee getallen linkt aan het hoofdobject moet je weten welke van de twee het aantal gebruikers is en welke het aantal bestelling.

Ook is het zo dat wanneer je deze twee tabellen aan elkaar linkt, dan krijg je de gegevens *verticaal*. Hier mee wordt bedoelde dat de attributen komen als nieuwe resultaten onder elkaar, per entity. Beter is het om de gegevens *horizontaal* komen te staan, een entity met de waarden als kolommen in de resultset. Dit zorgde er voor relatief slordige query's. (Zie Bijlagen C).

De gegevens kunnen elke vorm hebben op het moment dat ze vanuit de database komen. Dit zorgde er voor dat de gegevens extra goed gecontroleerd moesten worden bij de invoer.

Het schrijven van de query's was niet zo moeilijk, het bedenken van de query's wel. Het bijhalen van de correcte gegevens op de correcte plaatsen was nog wel ingewikkeld en soms best wel omslachtig. Het grote aantal JOINs en VIEWs was zorgelijk. Het leek overdreven veel werk voor kleine stappen, dat slechte performance zou geven op grote data sets.

7.2.6 ArangoDB

ArangoDB koste veel tijd om op te starten. Als eerste bleek dat de Work Station moest worden geüpgraded. ArangoDB werkt alleen op PCs met een 64 bit Operating Systems en er werd gebruikt gemaakt van een 32 bit. Gelukkig was er alleen een software update nodig. Het koste wel weer een dag om alle programma's opnieuw te installeren en alles weer recht te zetten.

Voor het query'en van de een ArangoDB Database wordt gebruik gemakt van AQL, Arango Querying Language. Dit verschilt veel met SQL en was moeilijk te leren. Gelukkig was de documentatie duidelijk. De uitleg was goed en er waren een aantal te volgen tutorial. AQL, de querying language van ArangoDB, heeft trekjes van SQL, maar is net wat anders gestructureerd. Het heeft veel functionaliteit, maar het duurt even voordat je door hebt wat je er precies allemaal mee kan, en hoe dat dan werkt. Hier is een voorbeeld van de verschillen tussen AQL en SQL:

```
SELECT * FROM users
  WHERE active = 1
  ORDER BY name, gender;
```

```
FOR user IN users
  FILTER user.active == 1
  SORT user.name, user.gender
  RETURN user
```

Figure 6: Voorbeeld verschil SQL en AQL. Afkomstig van: https://www.arangodb.com/why-arangodb/sql-aql-comparison/

Deze query's halen alle gegevens van de Table/Collection 'users' waar de waarde 'active' gelijk is 1 en gesorteerd op 'name' en 'gender'. Zoals je ziet is het verschil niet heel groot. Bij ArangoDB wordt er een loop gelopen door de Collection 'users' en wordt er met FILTER de WHERE van SQL uitgevoerd. Op dit niveau lijkt het makkelijk, maar als je wat dieper wil met AQL komen er snel andere dingen kijken wat verwarrend is. Het is vooral verwarrend omdat het in het begin zo lijkt op SQL, en daarna een stuk minder.

```
FOR u IN users
  FILTER u.id == @id && u.name == @name
  RETURN u
```

Figure 7: AQL Query met Bind Parameters. Afkomstig van: https://docs.arangodb.com/3.4/AQL/Fundamentals/BindParameters.html

AQL heeft ook een handige functionaliteit waar je gemakkelijk variabelen kan definiëren in de query. Bij het uitsturen van de query kan je een HashMap toevoegen waar je deze waarden in definieert. ArangoDB pakt dan de correcte gegevens en voegt ze toe in de query waar nodig.

```
1  for r in ECManage_Gegevens
2  filter r.Ronde.year == @year && r.Ronde.month == @month
3  for v IN 0..12 outbound r ECM_Rapport
4  return v
5
```

Figure 8: AQL Query met een Graph

Deze query maakt gebruikt van de Graph model van ArangoDB. In een 'Edge-Collection' kan je twee 'Documents' aan elkaar linken. Een wordt gelinkt als '_to' en de ander als '_from'. Deze Edge Collection kan je gebruiken om verschillende documents aan elkaar te linken die bij elkaar horen. In deze query wordt de specifieke record die voldoet aan de Bind Parameters eruit gehaald, en daarbij de 12 records die aan elkaar gelinkt zijn.

AQL heeft veel functionaliteit zoals te zien. Maar het is niet makkelijk om snel op te pakken. Dit komt voornamelijk doordat het veel op SQL lijkt, meet het niet is. Ook is de opslag methode anders dan in SQL. De JSON-Bestanden zorgen voor een grote vertaal slag die een tijdje kan duren voordat die valt.

Het proof-of-concept bleek uiteindelijk goed te werken en veel gebruikt te maken van de voordelen van noSQL over MySQL. Ook waren er verschillende onderdelen van ArangoDB die nog niet waren gebruikt, maar de database nog beter kon maken (Foxx Micro Servers, indexes en views).

7.2.7 EAV (Key-Value):

De EAV proof-of-concept kwam na het maken van de RDB en de noSQL. Dit model was een extra controle om te kijken of een Key-Value structuur ook een goede oplossing kon zijn. Het had de functionaliteit van noSQL, maar werkte op een SQL server. De gevonden Paper van de twee studenten (Lekberg & Danielsson, 2013) gaf duidelijke uitleg van hun aanpak. Dat zag er als volgt uit:

Figure 2.3: UML diagram of our database design

Figure 9 UML Diagram EVA Database (Lekberg & Danielsson, 2013)

- Van een record worden alle waarden opgesplitst en stuk voor stuk ingevoerd.
- Elk waarde kijkt eerste in de dbo.meta Table om te kijken wat het datetype is van het waarde.
- Er wordt een value-ID gemaakt voor de waarde, en die wordt, samen met een entity ID van de record, opgeslagen in de dbo.eav Table.
- En die zelfde entry in de dbo.eav Table wordt ook de naam van de Table waar de waarde opgeslagen moeten worden, behorend aan zijn data-type. Dit wordt gedaan door de naam van de Table op te slaan in het 'datya_table' kolom van de dbo.eav Table
    - Dit is omdat de studenten geen Foreign Key Constraints wilden invoegen in de database. De waarden kunne verschillende types hebben die je van te voren niet weet, dus heb je meerdere 'attribute' tabellen nodig. Deze kun je niet allemaal linken aan de zelfde tabel, omdat je dan niet kan garanderen dat alle gegevens opgeslagen worden.
- De waarde wordt aan de hand van zijn valueID in dbo.eav opgeslagen in de corresponderende 'attribute' Table.

Met dit ontwerp kan elke record opgeslagen worden, onafhankelijk van formaat. Wel is het erg omslachtig, dat gaven de schrijvers ook al toe:

> *"In contrast to this, our prototype is far more generic and can be used to different systems, but also means that the queries against the database should be designed a certain way, which results in a performance loss compared to the customized database. The reason for this being that our database prototype might require more operations for a single operation. However, if this loss is only small and acceptable, our generic database offers a quick way of implementing a database"* (Lekberg & Danielsson, 2013)

Het was erg ingewikkeld om gegevens correct in te voeren. Dit was omdat het linken aan de goede attribuut tabellen ingewikkeld was. Lekberg & Danielsson hadden hunzelf de restrictie opgelegd om geen Foreign Key Constraints te gebruiken naar andere tabellen. Dit betekende dat de naam van de Table waar de waarde ingevoerd moet worden, opgehaald moet worden met een query en dan doorgeven als een variable. Maar in SQL kan je de Table naam in een query niet als een variable doorgeven, omdat het programma niet weet of die naam correct is of niet.

Na verder kijken kwamen ook de volgende vragen op: Hoe wordt een hiërarchie gedefinieerd in dit systeem? Kunnen de gegevens van verschillende rapportage in de zelfde dbo.eav Table? Hoe worden verschillende entiteiten aan elkaar gelinkt?

Deze oplossing bleek niet het beste, maar het slechtste van beide werelden te zijn. Werkbaar in een SQL database, maar gebruikt geen van de functionaliteiten. Niet gebonden aan een vorm, maar toch moeilijk in te voeren.
Uit eindelijk is besloten om geen proof-of-concept van dit ontwerp te maken. Het ontwerp leek onhandig, niet bruikbaar op grote schaal en onnodig ingewikkeld.
Het kon zijn dat het proces onhandig is aangepakt en dat er wel een handige manier is van de database querying op deze manier. Het kan zijn dat er nog een andere model is van een Key-Value database in SQL die handiger werkt dan deze, maar niet is gevonden in het onderzoek. Maar omdat dit al een extra implementatie was, en de andere implementatie goed werken, is deze niet afgemaakt.

7.2.8 Presentatie

Nadat de twee proof-of-concepts af waren, is een gehouden voor de begeleiders, en hebben we samen gekeken naar de beste oplossing. Beide proof-of-concepts zijn met elkaar vergeleken op de beschreven vlakken. Zie Bijlagen D voor de slides van de presentatie.

De presentatie werd goed ontvangen, maar zorgde ook voor veel feedback. Sommige onderwerpen waren niet objectief genoeg behandeld of hadden nog wat stekende fouten in het ontwerp.
Uiteindelijk viel de keuze op ArangoDB. Dit was om de volgende redenen:

- Door het feit dat ArangoDB schemaless is, kan het bijna alle vormen van data tot zich nemen. Dit maakt het makkelijk om nieuwe bronnen toe te voegen
- Het is eleganter dan de SQL database, doordat in zowel invoer als uitvoer
- Het is leerzamer voor mij en voor OIS. Het ontdekken van een nieuwe tool en kijken wat het kan en hoe het werkt kan is fijn voor OIS.

7.2.9 Matrix

Hier onder een Matrix met het overzicht van de eisen en de methoden. De EAV Methode is uitgesloten omdat er geen vertical prototype is gemaakt.

Eis	noSQL (ArangoDB)	RDBMS (MariaDB)
Makkelijke om nieuwe data op te nemen	Zeer goed	Matig
Makkelijk om data uit te halen	Voldoende	Matig
Sterke Aggregeer methoden	Goed	Goed
Mogelijkheid voor verschillende datasoorten	Zeer goed	Voldoende
Sluit het goed aan bij OIS	Slecht	Goed
Makkelijk te leren	Voldoende	Goed
Uiteindelijke Keuze:	noSQL (ArangoDB)	-

7.3 Vertical Prototype

De volgende stap in het ontwikkel proces was het maken van een vertical prototype. Het doel van de vertical prototype was het ontdekken van alle hindernissen in het proces. Door het alle stappen van de applicatie een keer door te lopen kom je er achter wat er precies moet gebeuren en wat daar bij komt kijken. Met de kennis van de gemaakte vertical prototype kan daarna een nieuw ontwerp gemaakt worden waar alle grote problemen al zijn opgelost of ontweken. De vertical prototype is niet bedoeld om direct over te nemen naar de uiteindelijke applicatie, maar meer een leer proces.

De volledige applicatie kan zo worden gezien. De vier lagen die eerder uitgelegd zijn in het verslag zijn hier uitgebeeld. Er was de keus om een 'Laag' te ontwikkelen en daarna nieuwe lagen aan toe toevoegen, maar dit leek minder leerzaam. Niet alle onderdelen van een laag worden gebruikt, en het kan zijn dat bij het ontwikkelen van een latere laag ontdekt wordt dat er een fout is gemaakt die dan moeilijk is te herstellen.

Input				
.xml	Database Query	Html	Mail attachment	Nieuwe Bron
Gegevens Opslaan in database				
Maken overzichten				
Logic 1	Logic 2	Logic 3	Logic 4	Logic N
Maken Rapportage				
.pdf	.docx	Html	Bi tools	Overig
Versturen gegevens				
Online Web	Email	Zabbix	Post	Overig

De 'Kolom' waarvoor is gekozen gaat als volgt:

- Input: XML Bestand
- Maken Overzichten: ECManage
- Maken Rapportage: .pdf
- Versturen Gegevens: email

Ik heb voor deze kolom gekozen omdat die ook gebruikt wordt op het moment van ontwikkelen. Alles dat hier geleerd wordt zal ook belangrijk zijn om te implementeren in de uiteindelijke applicatie. Ook zijn sommige onderdelen ingewikkelder dan andere opties in de zelfde laag. Het maken van een .pdf ingewikkelder is dan een .docx of een html bestand, omdat daar nog extra tools bij komen kijken.

Aan het eind van de vertical prototype moet de input bestaan uit een mail attachment van een XML bestand. De applicatie moet zien dat hij een mailtje krijgt, en dan zelf alle stappen doorlopen met het ontvangen mailtje. Dit is nog een extra stap, maar is het doel van de uiteindelijke applicatie. Zo kan ook getest worden hoe modulair de applicatie is, en hoe makkelijk dingen kan aangepast kunnen worden in een laag.

7.3.1 XML Bestand

Iedere maand wordt er een mailtje gestuurd van uit de Oracle Database van OIS met twee .XML bestanden. Deze bestanden worden gebruikt om de huidige ECManage rapportages te maken. De XML bestanden worden direct vanuit de Oracle Database gemaakt, en hebben dan ook een database format. Dat ziet er als volgt uit:

Het bestand wordt aangeleverd als een ROWSET, met elke entry als een ROW. Elke ROW heeft de zelfde velden. De eerst entry is altijd een 'Default' entry. In de zelfde query krijgen we de Leveranciers en de Omgevingen van de Leveranciers terug. Dit moet verwerkt worden als een hiërarchie. De Omgevingen horen bij de Leverancier die er boven staat, totdat dat een volgende Leverancier komt. Je kan zien dat een entry een Leverancier is als het veld "Klanten" leeg is. Wat wel opvallend is dat Leveranciers zelf ook nog dingen kunnen bestellen en actieve mensen hebben.

```
<?xml version="1.0"?>
<ROWSET>
 <ROW>
  <ID>0</ID>
  <Leveranciers>Default</Leveranciers>
  <Portal>Default</Portal>
  <Aanmaakdatum>08/Apr/2011</Aanmaakdatum>
  <Nr_Actief>0</Nr_Actief>
  <Nr_Inactief>0</Nr_Inactief>
  <Jaar>2019</Jaar>
  <Maand>01</Maand>
  <Klanten/>
  <Nr_Orders>0</Nr_Orders>
</ROW>
<ROW>
  <ID>1</ID>
  <Leveranciers>Simple Workwear</Leveranciers>
  <Portal>SIMPLE</Portal>
  <Aanmaakdatum>08/Apr/2011</Aanmaakdatum>
  <Nr_Actief>35</Nr_Actief>
  <Nr_Inactief>5</Nr_Inactief>
  <Jaar>2018</Jaar>
  <Maand>05</Maand>
  <Klanten/>
  <Nr_Orders>5</Nr_Orders>
</ROW>
<ROW>
  <ID>2</ID>
  <Leveranciers>Simple Workwear</Leveranciers>
  <Portal>PRXS</Portal>
  <Aanmaakdatum>08/Apr/2011</Aanmaakdatum>
  <Nr_Actief>35</Nr_Actief>
  <Nr_Inactief>5</Nr_Inactief>
  <Jaar>2018</Jaar>
  <Maand>05</Maand>
  <Klanten>Praxis</Klanten>
  <Nr_Orders>5</Nr_Orders>
</ROW>
</ROWSET>
```

Figure 10: Format van XML Invoer

Als eerst werden een aantal Java Classes gemaakt die gebruikt konden worden om XML in te lezen. Deze objecten konden dan in de gewenste structuur overgezet worden naar de noSQL database van ArangoDB.
Na een dag klussen bleek dit best wel omslachtig was. Er moest veel converteer werk gebeuren om deze XML input in het formaat te krijgen zoals gewild, en vervolgens ook goed uitlezen zodat ze opgeslagen konden worden. Zodra de gegevens waren opgeslagen in de database, waren de ingevulde object niet meer nodig. Deze objecten konden weer gebruikt worden om het uit lezen, maar dat was ook maar kort nodig.
De begeleider was het er mee eens dat het omslachtig was. Hij vroeg zich af of het niet mogelijk was om de XML direct om te zetten in een JSON bestand, en die in de noSQL database te zetten. ArangoDB leest nu eenmaal JSON, en het verschil is niet zo groot tussen de twee documenten.

Dit was een van de grote leermoment van de stage. De ontwikkelmethode van OIS werd in een keer beter duidelijk. Wat er gemaakt was, was voor die specifieke invoer. Specifiek het formaat van de ECManage gegevens, en hoe die in de database moesten staan. Maar zodra een ander XML bestand ingelezen moest worden, moest daar weer opnieuw voor gemodelleerd worden. Het werk was niet herbruikbaar. OIS maakt dingen nu juist op de herbruikbaarheid en groei van software, en dat niet wat er gemaakt was. Er moest geen ECManage-XML-Inlees-module gemaakt worden, maar een Oracle-XML-inlees-module. Op deze manier konden eventuele andere XML bestanden uit de Oracle database ook ingeladen worden met de zelfde module en hoefde er geen extra werk gedaan te worden. Op deze manier is er wel minder invloed op hoe de gegevens worden opgeslagen in ArangoDB, maar dat is nu juist waar noSQL voor is. Gegevens worden snel opgeslagen, daar kunnen later overzichtelijke aggregaten van gemaakt worden.

Dit moment was een enorm leer moment over hoe er nagedacht wordt over ontwikkelen. Kijk naar de 'Bigger Picture' en naar de toekomst. Een keer 500 regels code schrijven is beter dan iedere keer 200 regels code schrijven.

7.3.2 PDF Generatie

Voor PDF generatie is er gebruik gemaakt van Flying Saucer. Na wat onderzoek, bleek Flying Saucer de beste oplossing. Tijdens het zoeken viel het met op dat sommige van de libraries wel 10 jaar oud waren. Het was een risico om deze libraries te kiezen, omdat er een verwachting was dat er al een beter oplossing was bedacht voor het probleem sinds dat het is uitgebracht.

Deze tool werkte erg goed. Het koste even om te leren hoe het gebruikt moest worden, maar daarna werd het erg makkelijk. Thymeleaf was erg interessant omdat het bij het Spring Boot project hoort. Een platform die helpt bij het maken van Java Projecten en het kiezen van Libraries. Spring Boot was nog niet eerder langsgekomen in het onderzoek, maar leek erg interessant voor verdere stappen.

7.3.3 E-mails versturen

Voor het e-mails versturen is gebruik gemaakt van de Java Mail Library. De Java Gmail Library is ook overwogen, maar liet snel hindernissen zien. Om het goed te testen heb zijn er twee nieuwe Gmail accounts gemaakt waar tussen e-mails verstuurd konden worden. Dit was om gevoelige informatie te beschermen.

7.3.4 Reflectie Vertical prototype

Het maken van de vertical prototype ging goed. Veel van de onderdelen waren al eerder getest om te kijken hoe het werkte. Deze stappen waren goed over te nemen. Ook is er veel geleerd over het ontwerp. De eyeopener voor het maken van een generiek XML reader gaf veel inzicht en was erg leerzaam. De korte tijd waarin de Vertical Prototype was gemaakt was een positief teken.

De presentatie van de prototype aan de stage begeleider ging ook goed. De begeleider was blij met de applicatie, maar had ook heel erg veel aanwijzingen. Hij ging gelijk nadenken over de grote schaal van de applicatie. Kijken hoe we het konden uitbreiden in dingen konden verbeteren. Hij maakte een lijst met vragen die had, die bij zo'n applicatie komen kijken. Hij wilde graag generieke oplossingen die veel gebruikt kunnen worden, zodat er 'zonder code schrijven een nieuwe bron, of output gemaakt kan worden'. Een grote uitdaging. Hij wilde ook graag meer duidelijkheid hebben, antwoorden op zijn vragen en diagrammen zodat hij makkelijk kon begrijpen hoe alles precies met elkaar samenwerkt.

8 Ontwikkelen Applicatie

Nadat alles was bewezen was het tijd om te beginnen aan de uiteindelijke applicatie. Er zijn veel grote beslissingen gemaakt: de database keuze, ontwerp keuzes en het bewezen met een vertical prototype. Met het vertical prototype zijn alle grote stappen als een keer doorlopen, en is dus duidelijke hoe het moet gebeuren.
De eerstvolgende stap was het maken van een UML Diagram zodat er duidelijkheid was hoe de applicatie werkt. De begeleiders kunnen er dan nog een keer naar kunnen kijken voordat het ontwikkelen begint.

8.1 Schrijven UML Diagram / Workflow

Deze stap was erg ingewikkeld. De aanwijzing van de stagebegeleider leken erg straight-forward, maar om het dan nog in elkaar te zetten bleek erg ingewikkeld. Als snel was het duidelijk dat er weinig ervaring was op dit gebied. Bij studie projecten wordt deze stap vaak overgeslagen en later afgeraffeld, maar niet was het een verplichte voordat er begonnen kon worden. De studie projecten zijn ook vaak erg gericht, de vraag is duidelijk en je kan de applicatie maken die aan die specifieke eisen voldoet. Op dit niveau moet er echt gekeken worden naar een grotere schaal. Deze applicatie moest voldoen alle gekregen eisen, en eisen die nog niet bepaald waren.

Het schrijven van de diagrammen was ingewikkeld. De aanwezig waren redelijk globaal en er was weinig ervaring met ontwerp op deze schaal. Vooral de vraag van uitbreiding “zonder code te schrijven” was een onduidelijk begrip. Een grote stap was het inbeelden van de werking van de verschillende systemen. Tijdens het implementeren van een applicatie wordt het ontwerp snel duidelijk doordat er een goed overzicht komt hoe verschillende systemen samenwerken. Het is moeilijk om dit in te beelden met weinig ervaring van ontwerpen.

Na een week aan werk is er nog een keer gesproken met de opdrachtgever om duidelijkheid te vragen. Het begrip “zonder code te schrijven”, maar niet letterlijk als werd bedoeld. De bedoeling stond voor het hergebruiken van modules voor meerdere functionaliteiten. Voor elke implementatie moet er natuurlijk wel code worden geschreven, maar als er geen nieuwe implementaties gemaakt hoeven te worden scheelt dat veel tijd.

Er werd op de goede manier na gedacht, maar het was minder ingewikkeld dan verwacht. Het was nog steeds wel moeilijk doordat er weinig ervaring was in het onderwerp.

8.2 Pakket Keuze

De ervaring met de REST API van ArangoDB, Foxx, bleek minder handig dan eerst leek. De REST Calls moesten in JavaScript worden geschreven, en elke aanpassing moest opgeslagen, gezipt en geüpload worden naar server.

Dit maakte het moeilijk om te debuggen en veel tijd om te ontwikkelen. Uit eerder onderzoek kwam Spring Boot naar voren, een Java Platform is veel verschillende Java Libraries aanbied. Spring Boot bied ook een sterke REST Service functionaliteit aan, maar geschreven in Java, en niet in JavaScript. Dit maakte Spring Boot een interessante keuze, en is er besloten in te verdiepen.

Spring Boot bleek ook een sterke integratie te hebben met MongoDB, de voornaamste noSQL Database model. Eerder is er gekozen voor ArangoDB als noSQL model, omdat de Querying Language krachtig leek, een multimodel systeem had en er mogelijkheid was voor JOINs.

Uiteindelijk is er in de vertical prototype niet heel erg veel gebruik van gemaakt. AQL Bleek soms ook meer een hindernis dan een hulp, zeker voor ontwikkelaars die nieuwe waren tot de taal. En de REST API Service maakte ontwikkelen moeilijk.

Na wat onderzoek naar MongoDB, bleek het verschil niet heel groot en leek het makkelijker in te stappen dan ArangoDB. De goede integratie met Spring Boot was ook een belangrijke factor omdat dat de nieuwe richting leek.

Omdat de voordelen van ArangoDB over MongoDB niet echt werden gebruikt, het ontwikkelen van de Foxx API zo ingewikkeld was en de goede integratie van Spring Boot met MongoDB is er gekozen om over te stappen naar MongoDB.

De nieuwe pakketkeuze, samen met de UML en Workflow diagrammen zijn gepresenteerd aan de opdrachtgever voor toestemming. Omdat de keuze goed was onderbouwd en de diagrammen uitgebreid waren was de opdrachtgever het er snel mee eens. De volgende stap in het proces was het ontwikkelen.

8.3 Ontwikkelen

8.3.1 Planning

De eerste stap in het ontwikkel proces was het maken van een Trello Board. Trello is een soort online White Board waarop taken geschreven en verdeeld kunnen worden. De grote taken zijn onderverdeeld in grote groepen en van de grote lijnen zijn onderwerpen gemaakt. Het Trello Board gaf een goed overzicht van de verschillende taken die gedaan moesten worden voor de applicatie.

Na het maken van het Trello Board is er een globale planning gemaakt van het project. Er zijn belangrijke milestone bedacht binnen het project en die zijn ingeschat op hun duur. De planning was globaal gemaakt met niet strikte lijnen zodat uitloop en koersverandering nog goed mogelijk was. De planning kwam er als volgt uit:

Onderwerp	**Deadline**	**Toelichting**
Website	10 Mei	Alle belangrijke features werken hebben op de site (Security, database connectie, aanpas functionaliteit en de lopende processen). Uiterlijk is nog niet belangrijk.
Alfa Build	6 Juni	Een volledig werkende versie hebben van de applicatie, die ook ingeladen is met de goede gegevens. De applicatie moet kunnen proefdraaien en dat op dat moment ook de goede dingen gebeuren. Ruimte voor bugs is nog toegestaan.
Documentatie/ Testen	28 Juni	De bugs moeten uit de applicatie zijn gehaald, en applicatie moet goed gedocumenteerd en getest zijn. Dit is de laatste stap in het traject en is een week voor het einde van de laatste deadline zodat er nog tijd hebt voor eventuele uitloop en het afronden van het verslag.
Verslag	5 Juli	Deze dag moet het verslag en de applicatie compleet af zijn.

Hoewel de planning erg globaal was, gaf het wel een goed beeld van wat te verwachten was. Op deze manier kwam er veel overzicht. De planning leek accuraat toen hij gemaakt werd. Onderwerpen waren goed uitgespreid, en er is tijd genomen voor de belangrijke taken.

8.3.2 Eerste Stappen

Het ontwikkelen begon met het maken van het Spring Boot Project. Spring Initializr biedt een handiger manier om een snel een project te maken met de gewenste dependencies. De gekozen dependencies, waarvan zeker was dat die gebruikt zouden worden, waren:

- Web
- Rest Repository
- Thymeleaf
- Security
- JPA
- MongoDB
- Mail

Daarna was het inrichten van het project. Dit was van te voren bepaalde in de UML diagrammen, en bleek redelijk goed te kloppen met wat handig was.

De schakel stap van MongoDB, en vooral de integratie met Spring Boot, was in het begin ingewikkeld. Gelukkig was er meer dan genoeg leermateriaal beschikbaar

Nadat MongoDB werkte in het systeem, begon het ontwikkelen van de website. De applicatie runt op een server, het moest nu eenmaal altijd bereikbaar en operationeel zijn. Als deze server toch al in de lucht is kan er net zo goed ook een dashboard zijn waarmee de applicatie te benaderen is. Op deze manier kunnen verschillende templates worden ingeladen en aangepast, nieuwe contact personen worden toegevoegd en geeft het een status van de gang van zaken.

De website was niet gemaakt in vertical prototype en was dus een nieuwe stap. In het begin was het vooral verschillende dingen uitproberen. Spring Boot heeft veel functionaliteit die dit goed aan de praat krijgt.

Spring Boot heeft een embedded server, die vanzelf aan gaat als de applicatie wordt gestart. De sterke Rest Service ondersteuning werkt net zo goed voor websites als voor API's.

Spring Boot bied ook uitgebreide ondersteuning voor Web Security. URL's kunnen gebonden worden aan bepaalde Rechten van gebruikers, en een standaard inlog methode wordt helpt met de verwerking.

Ook Thymeleaf, de HTML templater die gebruikt is voor het generen van de HTML templates van de rapportages, kan makkelijk gebruikt worden voor de website. De enorme flexibiliteit en de code die in de HTML zelf kan worden geschreven zorgde voor goede dynamische pagina's.

Voor de CSS is gebruik gemaakt van Bootstrap. Hier was al ervaring in, en geeft snel een responsieve uiterlijk dat er goed uit ziet.

8.3.3 Grafieken

In de ECManage Rapportages staan grafieken van de afgelopen tijd. Er is veel gekeken naar grafieken en de opmaak daarvan. Origineel is er gekeken naar Java gebaseerde Grafieken, dit was makkelijk in te stellen en kon 'offline' gegenereerd worden. Het probleem was wel dat deze grafieken dan moeilijk aan te passen, omdat de server gestopt moet worden, en vaak waren ze ook niet heel moei.

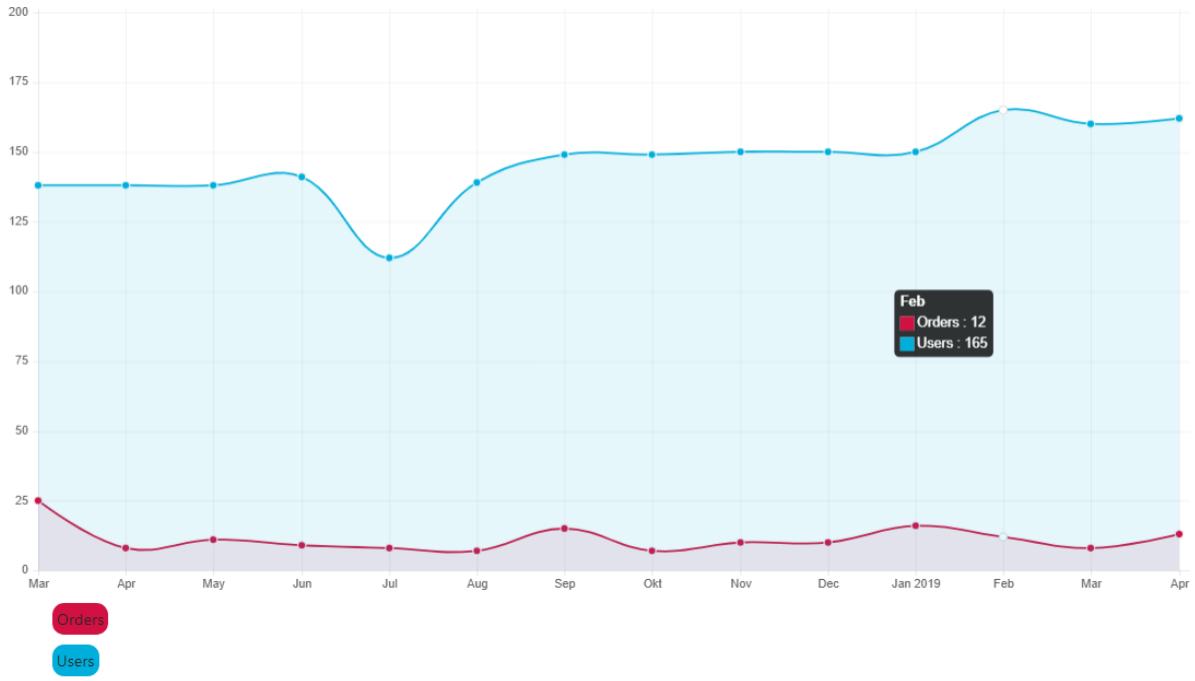
De Chart.js library maak mooie grafieken, maar die worden alleen gerenderd op Web Pagina's met de goede JavaScript. Wel zijn ze makkelijker aan te passen, door de gegevens in de JavaScript bestand aan te passen.

Origineel is er gebruik gemaakt van de ChartX, een Java Library is grafieken kan maken. Dit werkte prima, maar het resultaat was niet zo fraai. Na wat afweging en discussie met collega's is besloten te kijken naar de mogelijkheid om Chart.js te gebruiken.

Er zijn verschillende dingen geprobeerd met verschillende succes.

- Eerst is gekeken of de JavaScript direct gerund kon worden in Java. Java heeft drivers die JavaScript kunnen runnen. Door het gebruik van verschillende libraries werd dit te ingewikkeld om uit te voeren.
- Daarna is er gekeken naar een virtuele browsers in Java. Java imiteert dan een browser, die pagina's kan bezoeken. Er zijn een paar geprobeerd, maar meestal kreeg je hier alleen de html terug, en werd de Java Script niet uitgevoerd.
- Er is gekeken naar het maken van een Base64 String encoded van de grafiek. Op deze manier kon die makkelijk worden meer gegeven en weer terug worden gezet. Nu bleek het encoden en decoden redelijk veel tijd te kosten, en ging het ook te kort aan de kwaliteit van de afbeelding.
- Uiteindelijk is er gekeken naar Selenium, en test engine waarmee websites getest kunnen worden. Het opent een Browser en voert de aangegeven stappen uit. Hiermee kon er worden ingelogd en de pagina's worden gerenderd en de grafieken gedownload worden. Dit kost wel meer tijd, en voelt een beetje omslachtig, maar zorgde voor een goed resultaat.

De laatste zorgde voor het beste resultaat. Het was moeilijk te zeggen hoe het zou reageren op een grote schaal, en of het zou werken als het op de server runt, maar dat was voor een verder stadium om te kijken. Het was nu duidelijk dat het in ieder geval kon. Hier onder is een voorbeeld te zien van hoe de grafieken eruit zien.



8.3.4 Tweede Database

De gegevens die gebruikt moeten worden voor de IRIS Rapportages staan op een Oracle Database die is aangesloten op het OIS Systeem. De gemaakte applicatie had al een verbinding met een MongoDB Database. De twee verschillende soorten database bases (SQL en noSQL) moesten samen

in de zelfde applicatie naast elkaar kunnen werken. Spring Boot heeft een sterke ondersteuning voor deze samenwerking.

De gegevens worden met JPA opgehaald uit de Database, en gelijk in een model gegeten. Daarna kunnen deze modellen gelijk opgeslagen worden in de MongoDB Database. Een operatie van de gehele tabel te kopiëren naar de Database, kan in twee regels code. Dit was enorm handig.

Verder kunnen er ook methodes worden geschreven op de JPA Repository die gegevens met een SQL query uit de database kunnen ophalen.

Deze stap leek ingewikkeld, maar blijkt zeer gemakkelijkheid door de functionaliteit die Spring Boot biedt.

8.3.5 Aggregeren IRIS Meldingen

De gegevens die in de rapporten staan zijn geaggregeerde data. MongoDB is erg goed in het aggregeren van grote stukken data. Dit gaat via de MongoDB Pipeline. In stappen kunnen er specifieke handelingen worden uitgevoerd. De volledig aggregeerde data kan vervolgens opgeslagen worden in een andere 'Collection'. Deze nieuwe data records worden gebruikt om de rapportages te maken.

Een van de eisen is dat de rapportages flexibel en makkelijk aan te passen zijn. Hiermee wordt bedoeld dat, wanneer er een nieuwe kolom erbij moet, dat dat automatisch in de verslagen erin wordt gezet. Dit wordt aangepakt door de aggregaat functie los in een Text bestand te zetten. Wanneer het aggregaat aangepast moet worden, dan kan dit aangepast worden in het bestand. Hiervoor hoeft de server niet uitgezet te worden. De tabellen van de Rapportage worden gemaakt aan de hand van het aggregaat. Het aggregaat wordt op een specifieke manier opgemaakt, en met die gegevens wordt er een HTML tabel gemaakt. Dit gebeurt in een Java Module, en niet in Thymeleaf, zodat er meer controle over is. De module kijkt naar de velden van het aggregaat en maakt gepast daarop de tabellen en kolommen.

8.4 Inlezen van ECM Gegevens

De gegevens die in de ECM Collection kwamen te staan waren bewaard in .XML bestanden. Van elke maand, beginnend in oktober 2011, was er een .XML bestand met de benodigde gegevens. In het test proces waren een aantal test .XML bestanden gemaakt met test gegevens. Er was functionaliteit gemaakt om deze gegevens uit het bestand te halen en in de Database te zetten. Echter, bij het verwerken van de gegevens bleken er veel complicaties te zijn.

- Als eerste was de structuur van de XML een keer aangepast. Er moest een nieuwe inlees module gemaakt worden voor de andere structuur die nog niet verwerkt was.
- Voor deze periode werd er ook een los bestand bijgeleverd die de Orders van de Records los bij hield. Dit bestand moest ook uitgelezen worden en toegepast worden op de goede Records.
- Een tijd lang werden gegeven uit twee verschillende databases gehaald, met hier en daar gedupliceerde gegevens. Deze gegevens zijn gefilterd door te kijken naar de portaal code. Records waar de portaal begon met een 'x' kwamen uit de tweede database. Deze gegevens werden opgehaald en verwijderd na het opslaan van alle gegevens.
- Er bleken heel erg veel Test omgevingen te bestaan die niet meegenomen moesten worden bij de totale berekeningen. Deze omgevingen, en de bijbehorende leverancier, werd met expliciet niet in de aggregaat meegenomen.
- Veel omgevingen bleken niet meer actief te zijn. Dit kan voor problemen zorgen bij het ophalen van alle omgevingen, omdat je dan ook gesloten omgevingen erbij haalt. Na het

invoegen van alle records wordt gekeken naar de meest recente record. Als deze niet overeen komt de huidige gegevens, wordt er " - Inactief" achter de gegevens gezet om te marken. Hier kan ook op worden gefilterd bij het generen van de rapportages.

- Sommige omgevingen zijn van naam veranderd. De aggregatie gaat op basis van de naam, dit zorgde voor foute gegevens. Er is een methode geschreven die voor elke portal code, de meest recente naam pakt, en die toepast op alle records met dat portaal.
    - Nu kwam het ook voor dat sommige omgevingen opgehouden waren voordat de naam werd veranderd. Dit zorgde er voor dat de naam niet kon worden aangepast, omdat de meest recente naam de oude naam was. Bij deze items is er in de .XML bestanden de oudste handmatig aangepast naar de nieuwe naam.
- Verder bleken er ook veel fouten te zitten in het bestaande programma van de rapportages. Bepaalde omgevingen en leveranciers werden, zonder duidelijke reden, genegeerd in het aggregeer en rapporteer proces. Dit zorgde in grote verschillen tussen de gemaakte aggregaten en de bestaande oude geaggregeerde gegevens. Na veel onderzoek, aanpassingen en nakijk werk is de beslissing gemaakt dat het nieuwe aggregaat leidend werd, en de oude niet meer.

8.5 Samenvoegen van de elementen

Tot op dit moment waren alle elementen los van elkaar ontworpen en getest. De volgende stap in het proces was het samenvoegen van de verschillende elementen en de volledige stroming doorlopen. Dit was een grote stap omdat er veel verschillende processen waren:

- Inlezen en opslaan van de gegevens
- Aggregeren van de gegevens
- Aanmaken van de templates
- Generen van de Pdf's
- Pdf's versturen naar de klanten.

Het linken van de processen zorgde er voor dat er veel oude bugs en work arounds naar voren kwamen. Het fixen van deze gegevens was moeilijk en tijdrovend omdat er telkens het grote proces ervoor doorgelopen moest worden.

Maar veel van de stappen werkte snel op de gewenste manier. Ook werden er betere oplossingen gevonden voor de processen en werden er veel meer streamlined gemaakt.

8.6 Deployment

Nadat alle verschillende elementen samen waren gevoegd ging het traject door naar het Deployment. De applicatie moet gebruikt kunnen worden op OIS en op een van hun servers kunnen draaien, dan kan er niet een instantie van Eclipse draaien iedere keer.

Het hele project is er gebruikt gemaakt van Maven, volgens hun website een 'Software project management and comprehension tool'. Dit zorgde er voor dat de deployment fase redelijk makkelijk werd. Met een specifieke Maven command kon een 'war' (Web Jar) gemaakt worden. Na nog wat kleine aanpassingen (rerouting van bepaalde bestanden en toevoegen van specifieke libraries en dependencies) was de applicatie direct bruikbaar vanaf de command line.

De opdrachtgever had nog wat specifieke eisen voor bepaalde onderdelen die ook nog toegevoegd moesten worden. Al deze opties leken redelijk wat werk, maar door goed ontwerp en een hoop hulp van Spring Boot vielen al deze aanpassingen enorm mee.

- Salten van wachtwoorden
    - De wachtwoorden van de gebruikers moesten gesalt worden, een extra stap in het beschermen van wachtwoorden. Na wat prutswerk bleek de gebruikt wachtwoord Encoder (BCrypt), dit al van zelf te doen.
- Headless maken van Grafiek Generatie
    - In de applicatie werd, zoals eerder beschreven, de grafieken gegenereerd door met Selenium een Test browser te openen en specifieke URL's te bezoeken en een afbeelding te maken. Het openen van de browsers is best wel ingrijpen, maar op het moment van ontwikkelen was er geen beter oplossing te vinden. Nadat de opdrachtgever had gevraagd of deze functie headless kon, werd er meer onderzoek gedaan over die die term. Bleek dat er met een simpele regel code deze instelling aangezet kon worden, en de code verder goed bleef werken.
- Mail protocol van OIS werkt met IMAP
    - De applicatie werkte op dat moment nog met POP3S omdat daar de beste beschrijving van was te vinden. Deze functies moest omgezet worden naar IMAP, omdat de mailserver van OIS hier mee werkt. De email module bleek voor beide functionaliteit te bezitten, dus was het snel gefixt.

Deze .war moest op een remote locatie draaien. Op deze manier was het niet OIS's hun verantwoordelijkheid om de machine aan te houden en kan de server altijd draaien.

Opnieuw kwamen Maven en Spring Boot enorm te pas. De omgeving was van te voren door de systeem beheer ingericht en hield een aantal programma's, sommige waarvan helemaal niet nodig bleken te zijn. De installatie was best wel straight forward:

- De MongoDB Server moest geïnstalleerd worden en aangesloten op het Path zodat deze gebruikt kon worden vanuit de Console.
- De dump van de MongoDB Database moet worden overgezet naar het nieuwe systeem.
- De Oracle JDBC moet geïnstalleerd worden op het systeem. Oracle slaat zijn systemen niet aan op Maven, en kan daardoor niet direct worden meegegeven.
- De Target folder moet over gezet worden naar de remote Desktop.
- Het .war bestand moet gestart worden via console met een simpele functie.

Maven bracht alle dependencies zelf mee, en Spring Boot runt een eigen embedded Tomcat én MongoDB Server. Deze installatie ging, nadat alle stappen duidelijk waren, erg gemakkelijk.

Bij het test draaien van het systeem kwam een grote architectuur fout naar voren. De remote desktop, waar de applicatie draait, wordt gehost in een andere server. Mijn gemaakte applicatie maakt verbinding met de lokale Oracle Server. De externe server mag, door security redenen, geen verbinding maken met de lokale server. Dit betekend dat de IRIS Rapportage Functionaliteit, die de gegevens uit de Oracle Database nodig heeft, niet gebruikt kan worden. Dit is erg jammer, want de functionaliteit werkt. De helft van het project kan in deze staat niet gebruikt worden door miscommunicatie. Op een later moment moet een lokale server worden ingericht waar het systeem wel contact kan maken met de Oracle Database. De server kon verhuisd worden, maar volgende de opdrachtgever had de IRIS Rapportages geen prioriteit.

Dit had verholpen kunnen worden met duidelijk communicatie , waardoor de externe omgeving beter opgesteld kon worden. Wel is het zo dat de functionaliteit al is geïmplementeerd, dus dat het op een later moment verwerkt kan worden.

8.7 Overdracht

Nadat duidelijk was hoe het systeem geïnstalleerd moest worden, is de overdracht voorbereid. Er was een collega aangewezen, die zelf ook programmeert in Java, die de applicatie overneemt na de stage is afgelopen. Samen is er gekeken naar het opzetten van het project, en mogelijke problemen in het proces.

Eerst is een werkende build overgezet naar het workstation van de collega. Vanaf daar zijn alle onderdelen geïnstalleerd en het project gebuild zodat het werkte. Daarna is de code doorgelopen om te vertellen hoe onderdelen werkten. Het project is erg uitgebreid, daardoor zijn er een aantal dingen overgeslagen omdat de functionaliteit niet echt ingrijpend was voor de directe werking van de applicatie. Tijdens het doorlopen van de code kwamen er een aantal onderwerpen naar voren op het gebeid van Code Design. Een paar onderdelen waren niet handig ontworpen wat leidt tot restrictieve functionaliteit.

Andere onderdelen waren nieuw voor de collega. Hij had nog weinig ervaring met Maven, MongoDB en Spring Boot. Gelukkig was hij bereid om hier extra onderzoek naar te doen, maar dit zorgde ervoor dat de overdracht wel ingewikkelder werd.

Uit dit gesprek zijn nieuwe aandachtspunten opgesteld. Het doel was om deze, zo ver de tijd ons bracht, te implementeren of aan te passen:

- Overzetten van Git
    - OIS maakt zelf geen gebruik van Git. Het project werd gehost op een eigen private Git Repository. Er is besloten de Repository, nadat het project afgerond is, van de Git Repository af te halen klonen naar OIS hun eigen versie beheer. Dit maakt onderhoud voor hun makkelijk, ze hoeven geen Git te leren, en gaat de controle over naar de OIS.
- Meer instellingen in config files
    - Er zijn veel instellingen van het systeem die ingepast kunnen worden. Het is onhandig om voor iedere aanpassingen de server opnieuw op te starten en vervelender om de .war opnieuw te bouwen. Er wordt gekeken of een aantal van de instellingen (scheduled tasks, aggregaten, verbindingen) beschreven kunnen worden in een .config file zodat de aanpassingen live kunnen worden doorgevoerd.
- Errors handling
    - Veel van de Error Handling staat op een onhandige manier beschreven. Er staan nog veel Auto Generated code en vaak is er gekozen voor een e.printstacktrace(). Dit stopt de functie, terwijl het soms maar op een specifieke plek fout gaat. De Error Handling moet worden uitgebreid om het proces zo goed mogelijk te laten lopen, maar ook de beheerders zo goed mogelijk in te lichten over de errors.
- REST API
    - Veel van de Database is nog te benaderen vanaf de REST API. Dit is zo ontworpen zodat de functionaliteit handig te testen was. Die veel van de functionaliteit werkt, zijn deze Endpoints niet meer nodig. Samen is besloten veel van deze Endpoints te sluiten zodat nieuwsgierige gebruikers hier geen misbruik van kunnen maken. Sommige blijven open omdat er AJAX elementen zijn die ze gebruiken, en zodat eventuele nieuwe functionaliteit op deze manier kan worden getest en geïmplementeerd kan worden.

- Specifiek functionaliteit
    - De collega merkte correct op dat de rapportages altijd gemaakt worden op basis van de laatste gegevens. De nieuwe ECM Gegevens worden toegevoegd aan de database, en de aggregaten worden gemaakt op de meest recente gegevens. Maar dit betekend dat er geen overzicht kan worden gemaakt van de gegevens in het verleden, zonder een nieuwe aggregaat te maken. Als er een fout blijkt te zitten in de gegevens, kan deze wel gerectificeerd worden, maar kan er niet makkelijke nieuwe rapportages gemaakt worden. Er wordt gekeken of deze functionaliteit ingevoegd kan worden.
- IRIS Meldingen Rapportages
    - De IRIS Meldingen Rapportage Functionaliteit was dus niet inbegrepen in de uiteindelijke overlevering. Het is wel gemaakt en functioneel, maar moet later worden verwerkt. Deze code moet goed beschreven wordt zodat de beheerder het makkelijk kan overnemen in een later functionaliteit.

8.8 Afronden

Na het overdracht gesprek ging het project door naar afronden. In deze fase moet het systeem foutloos kunnen draaien, en de overdracht naar de opvolger goed in elkaar gezet worden. Door de tijdsdruk zijn er duidelijke prioriteiten gemaakt.

De hoogste prioriteit lag op het Error Handling. Er is veel functionaliteit in het systeem, en het is belangrijk dat het system er niet uit klapt bij een kleine fout. Vooral ook omdat de beheer zich niet honderd procent wijs is in de code. Er is gewerkt om de output zo overzichtelijk mogelijk te maken, en de meest voordehand liggende errors op te vangen. Een belangrijke feature die in deze fase is toegevoegd was Retryable. Dit zorgt er voor dat, wanneer een methode een error maakt, het systeem de methode nog een keer aanroept. Dit is gebruikt de email uitlezer, die niet zeker kan weten of de email al binnen is, snel nog een keer kan kijken of de email al is ontvangen. Dit was een goede oplossing voor een probleem dat makkelijk fout had kunnen gaan.

De tweede prioriteit lag bij de overdracht. Er is veel werk gestoken in het maken van een duidelijk overdrachtsdocument. Omdat de ontwikkelaars bij OIS nog niet echt bekend waren met bepaalde tools die gebruikt waren bij het systeem (Spring Boot, Maven, Git, MongoDB). Er is een duidelijk stappenplan, inclusief screenshots, gemaakt van het proces om de applicatie te kopiëren en te builen op een eigen workstation. Maar naast buiten het systeem, zijn er ook veel duidelijk comments in de code toegevoegd.

De derde prioriteit lag bij het maken van nieuwe functionaliteit. Na het maken van de uitgebreide Error Handling werd duidelijk waar grote pijn punten lagen in het systeem. Door de tijdsdruk was het niet mogelijk om grote delen van de applicatie om te schrijven. In plaats daarvan zijn er ‘patches’ gemaakt die wat struikel punten konden opvangen. Het herladen van de ECManage Database was daar een van. Eerst was hier een API Call voor in plaats, maar om de veiligheid van het systeem te bewaren zou deze worden dicht gegooid. In plaats daarvan kwam er een knop op het Admin Overzicht om de database te resetten. De functionaliteit was er al, het hoeft alleen omgezet te worden. De oude reset functionaliteit duurde lang, rond de vier minuten. Dit was omdat alle fouten uit het XML systeem iedere keer rechtgezet moesten worden. De oplossing hiervoor was het maken van een uitdraai de ECManage Database in goede staat, en deze inladen. De scheelde heel erg veel priegel werk en moeilijke stappen. Er hoefde geen aanpassingen gemaakte worden aan de bestanden, omdat die al goed waren. Ook verlaagde dit de duur van het reset naar ongeveer anderhalve minuut.

8.9 Presentatie

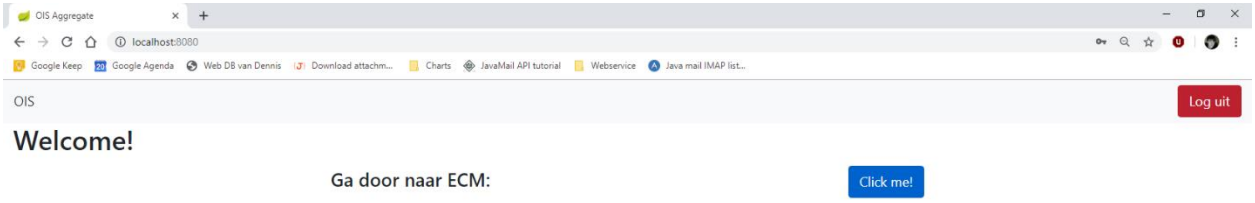
De laatste week van het project was vooral gericht in het paraat maken van de van het project voor deployment en overdracht. Er is expres niet te veel gewerkt aan het maken van nieuwe features omdat we bang waren dat het voor grote bugs zou werken. Ook is er een presentatie voorbereid voor de collega's. Na 5 maanden waren de meeste wel benieuwd naar wat er precies werd gemaakt. En is vlot een presentatie in elkaar gezet met een rondleiding door het systeem en een uitlichting van wat nieuwe technieken.

9 Project Rondleiding

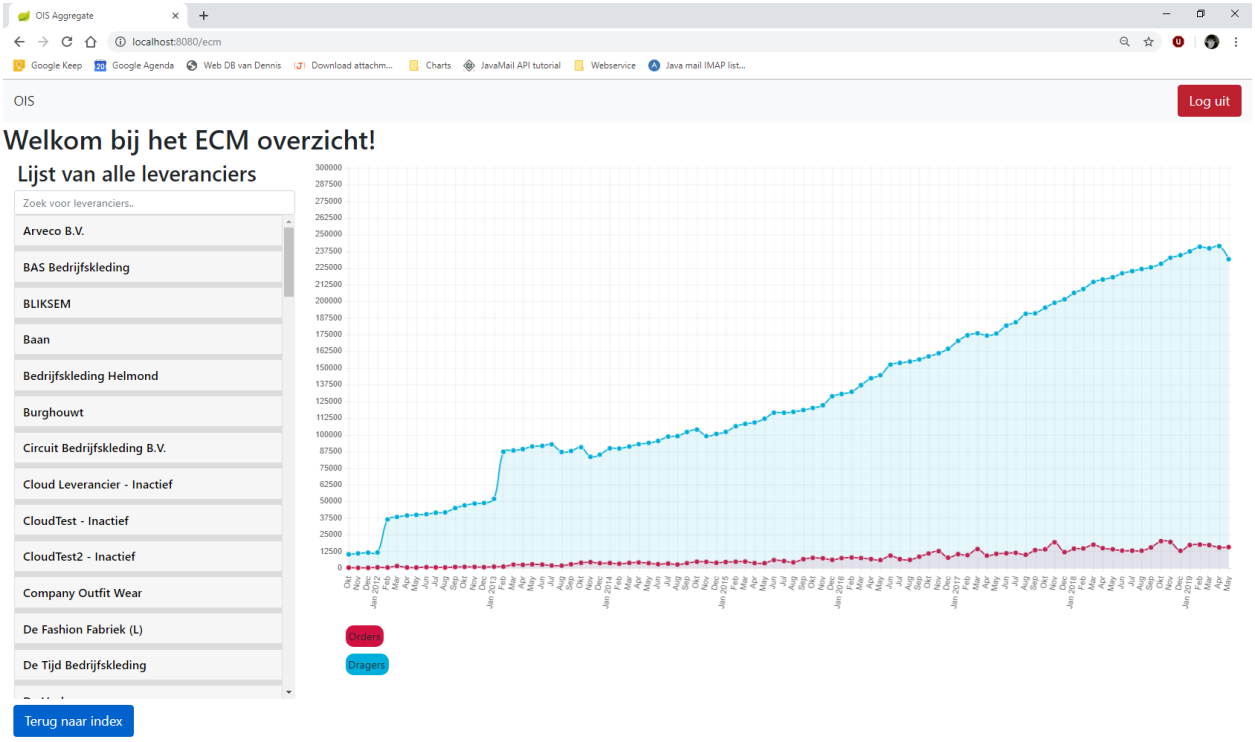
9.1 ECManage

Het systeem loopt zelfstandig, en heeft weinig hulp nodig. De ECManage beheerder is verantwoordelijk voor het toevoegen van de contact personen, instellen van de taal en de het schrijven van opmerkingen.

1. Ga naar de {URL}
2. Log in met de gebruikersgegevens.
    a. Neem contact op met de ADMIN voor het aanmaken van gebruikers gegevens

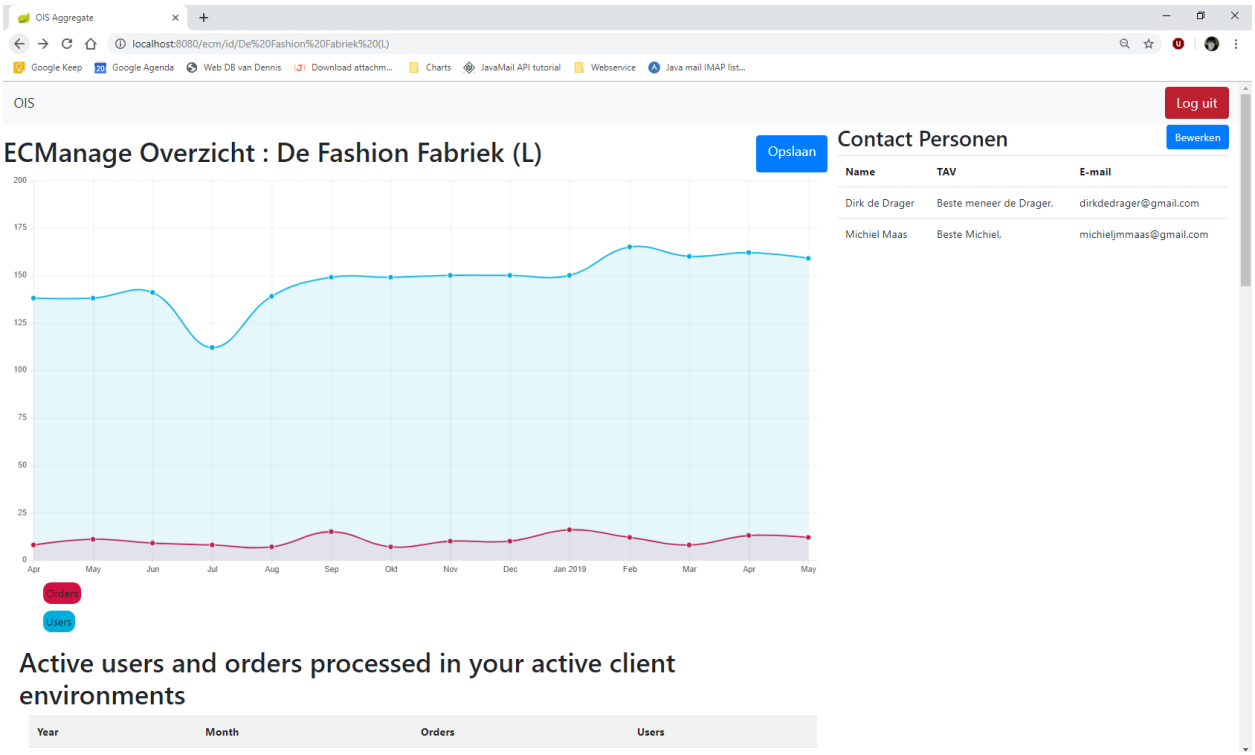


3. Klik op ECManage

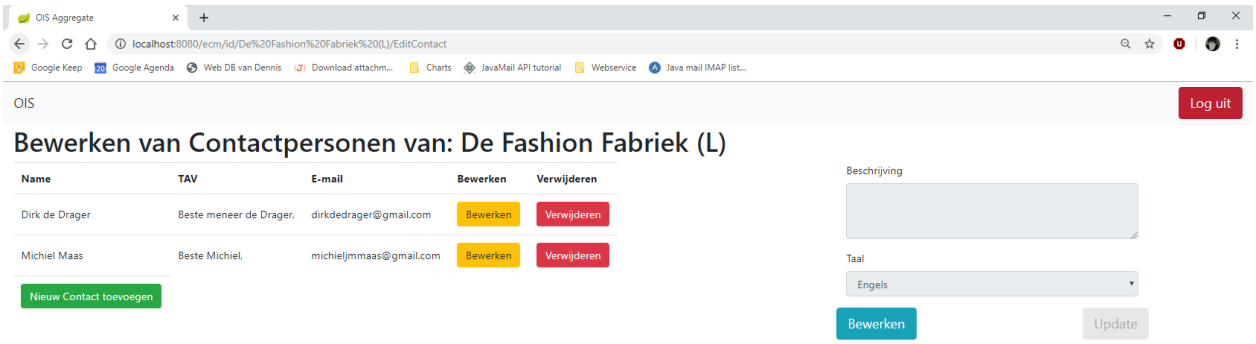


4. Dit is het overzichtsscherm. Dit geeft een overzicht van alle omgevingen over de loop van de tijd.
    a. Het linker zoek veld helpt je bij het vinden van de goede omgeving

b. Omgevingen beschermen met " - Inactief", zijn niet meer aangesloten op het ECManage systeem.



5. Dit scherm geeft een overzicht van de huidige omgeving, en de laatste 14 maanden. Het laat een grafiek zien met de Orders en de Dragers, een tabel van de afgelopen 14 maanden met de gegevens, en een tabel met de Orders en Dragers van de omgevingen van de afgelopen maand.
    a. Deze gegevens staan in het Engels, omdat de rapportages ook in het Engels worden gegenereerd.
    b. De Opslaan knop genereert de rapportage van de huidige omgeving.
    c. Rechtst staan de contact personen. Deze mensen krijgen iedere maand een mail met de rapportage. Klik op bewerken om hun gegevens aan te passen, of iemand bij te voegen / verwijderen.
6. Bewerken scherm geeft een overzicht van alle contact personen van dat bedrijf. Hier kunnen



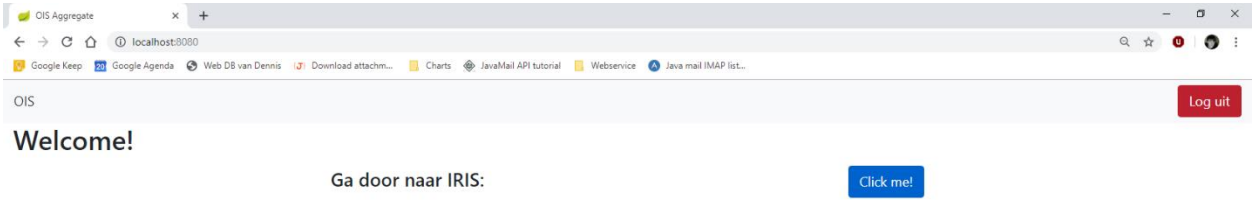
nieuwe contact personen worden toegevoegd, bestaande worden verwijderd of bewerkt. Ook is er een optie op de gegevens van het bedrijf aan toepassen.

a. De taal functie bepaald in welke taal de email wordt verstuurd. De Rapportage zal altijd in het Engels zijn.
b. Het beschrijving veld geeft de optie om een notitie toe te voegen aan de rapportage. Bij het leeg laten van het veld komt er “This week, no additional comments” te staan.

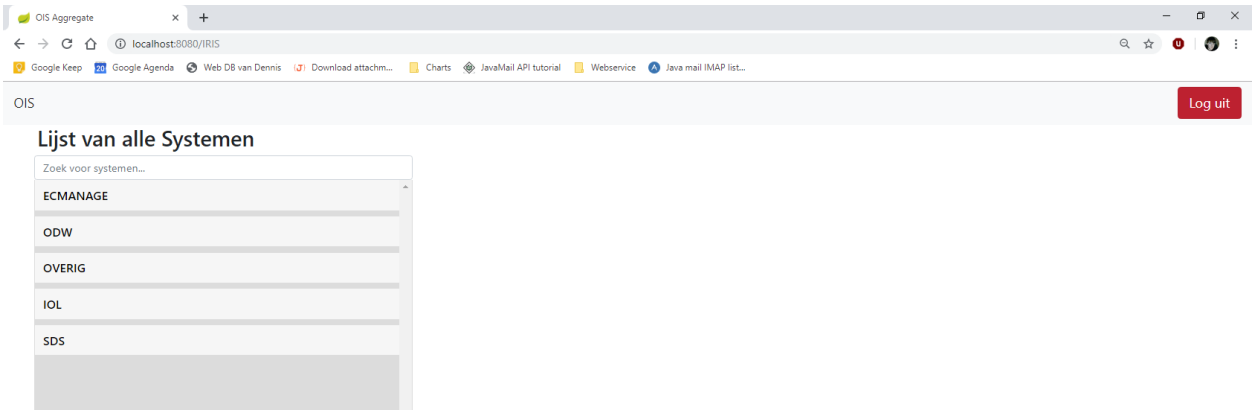
7. De eerst van de maand stuurt het systeem automatisch de rapportages naar de actieve omgevingen.

9.2 IRIS Meldingen

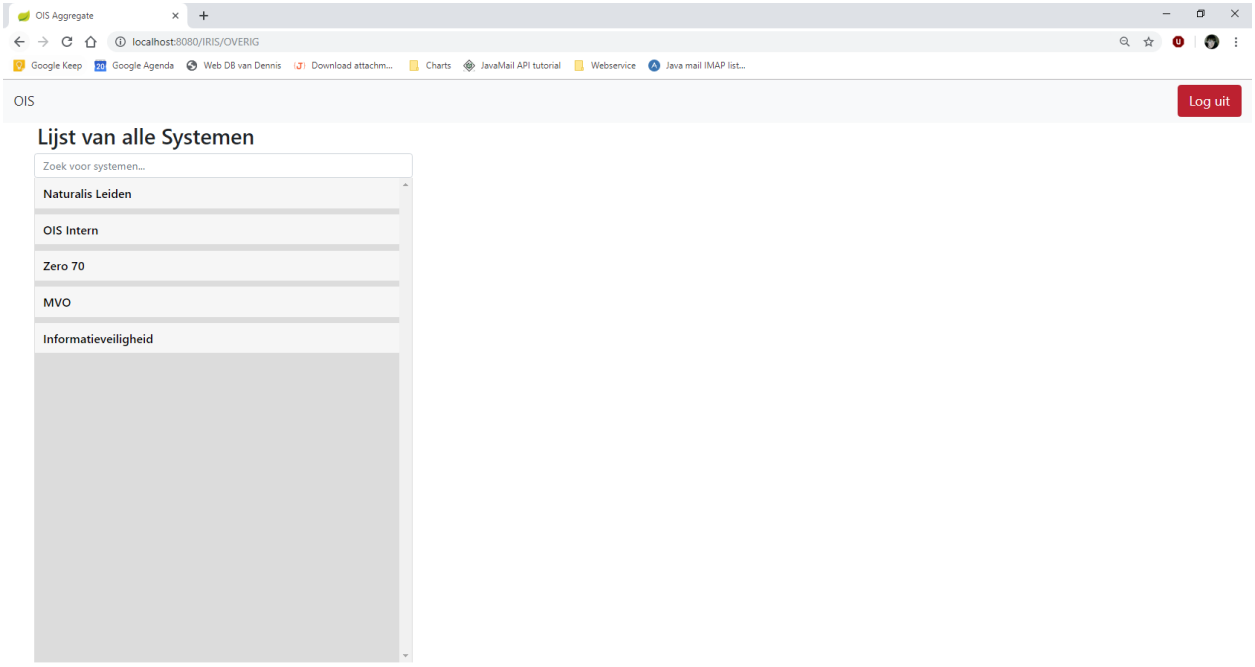
1. Ga naar {URL}
2. Log in met gebruikersgevens.
    a. Neem contact op met de ADMIN voor het aanmaken van gebruikers gegevens
3. Klik op IRIS



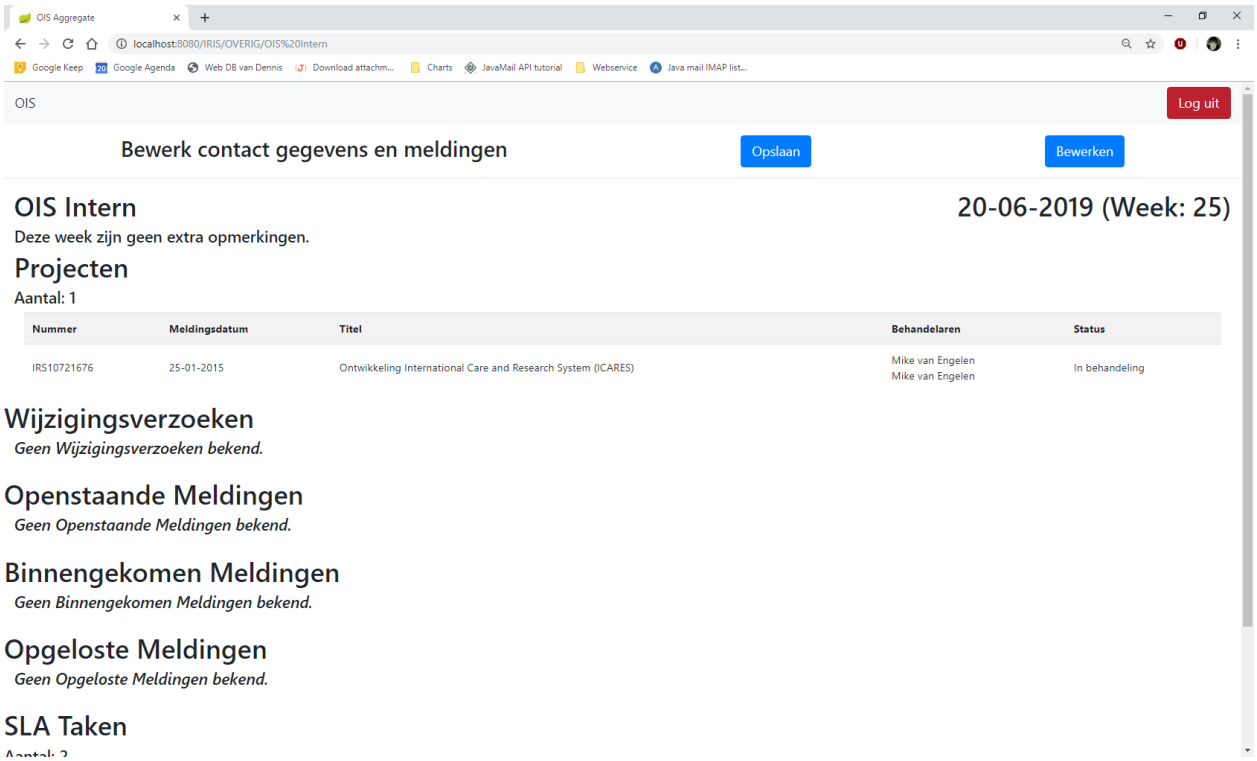
4. Klik op de omgeving waar de klant bij hoort.



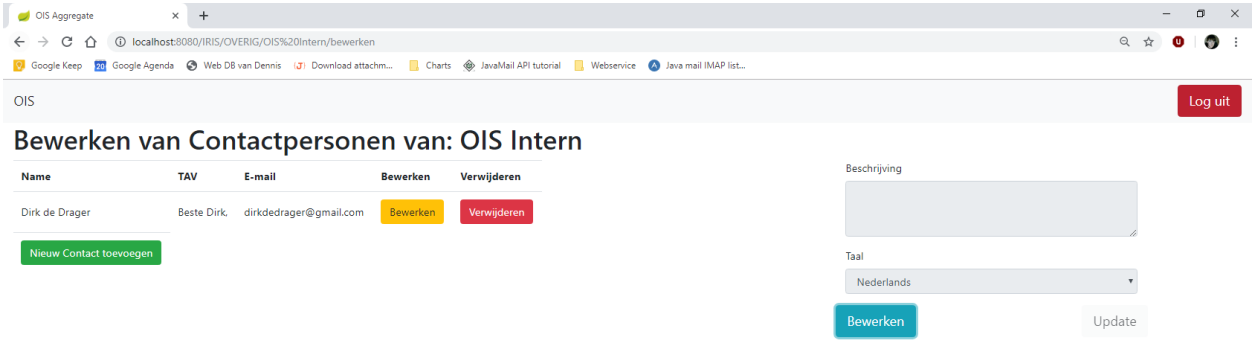
5. Klik op het bedrijf. Gebruik het zoek veld om de goede klant te vinden.



6. Dit is het overzicht scherm. Dit laat de meldingen zien de laatste keer zijn opgehaald (Ieder maandag om 0:00).
    a. Als je op opslaan klikt maakt het een rapportage van de gegevens die er op dat moment staan, en wordt die gedownload.
    b. Bij bewerken krijg je inzicht op de mensen die de contact gegevens krijgen.

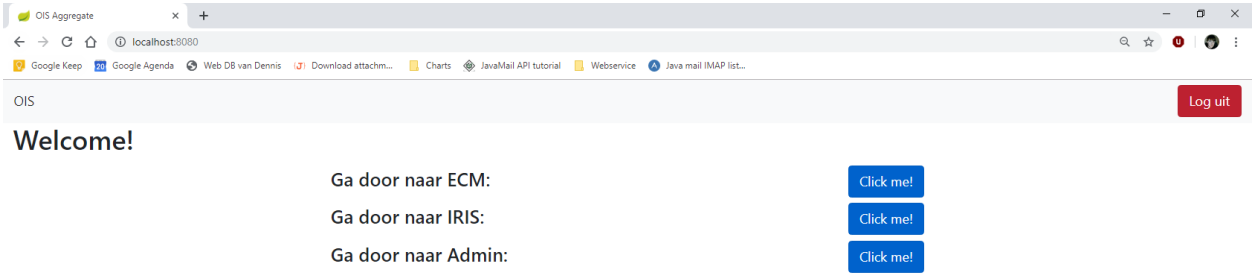


7. Dit overzicht laat zien wie er allemaal een mail krijgt van dit overzicht. Hier kunnen nieuwe contactpersonen worden toegevoegd, en bestaande bewerkt of verwijderd worden.
    a. Recht kan ook een beschrijving worden toegevoegd bij het verslag. Als dit veld wordt leeg gelaten komt er ‘Deze week geen opmerkingen.’ te staan. De taal kan ook worden aangepast, maar dit veranderd alleen de taal van het mailtje dat verstuurd wordt. De rapportage blijft in het Nederlands.

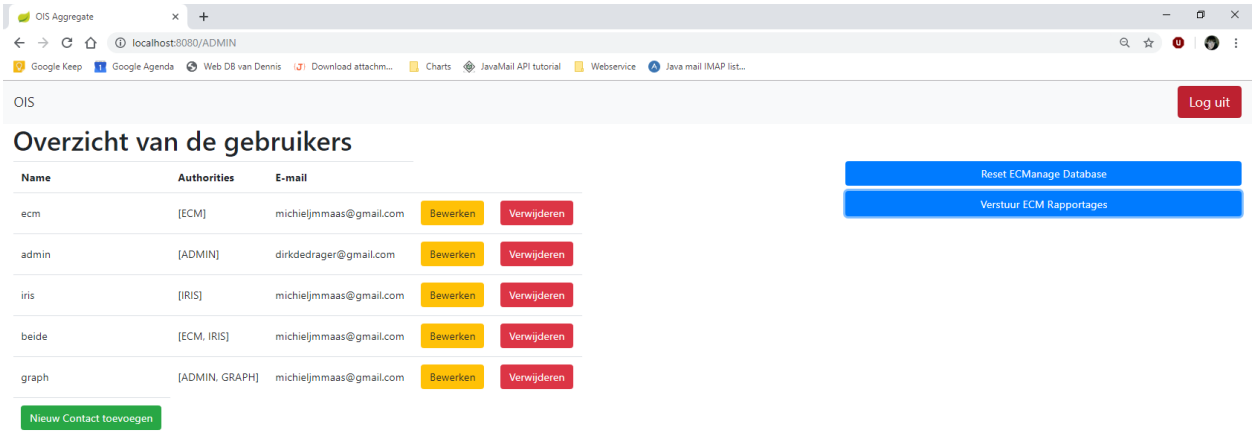


9.3 ADMIN

1. Ga naar {URL}
2. Log in met de gegevens.
    a. De ADMIN Maakt de gegevens aan



3. De ADMIN rol heeft toegang tot alle opties, zie vorige beschrijvingen voor handleidingen. Klik op ADMIN voor de ADMIN Instellingen.

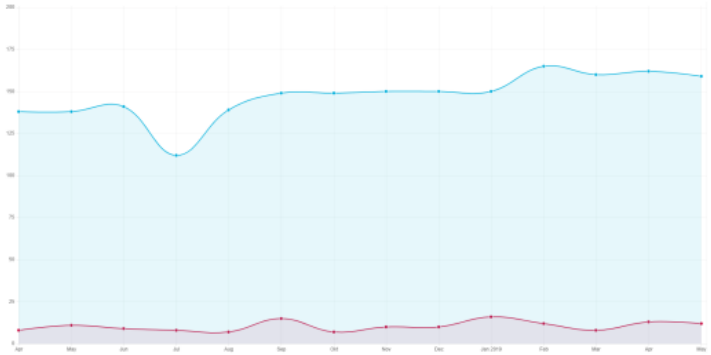


4. Hier kunnen nieuwe gebruikers worden toegevoegd en hun Autoriteiten worden ingesteld.
    a. Email wordt niet gebruikt
5. Met de ‘Database resetten’ knop kan de ECManage database worden gevuld worden met de gegevens uit het dump.xml bestand, en de xml bestanden uit de xml/Launch directory.
    a. Als een keer de berekeningen fout gaan kan bestand die de xml/Launch folder worden gezet, en de reset knop worden ingedrukt.
6. Met de ‘Verstuur ECM Rapportages’ knop kunnen er nog handmatig rapportages worden verstuurd. Van alle klanten worden de nieuwe rapportages aangemaakt en naar de Gmail. Het maakt gebruikt van de huidige gegevens in de database.
7. Autoriteiten
    a. Admin rechten over alles
    b. ECM : Rechten over ECM
    c. IRIS: Rechten over IRIS
    d. Toegang tot de pagina’s van de grafieken

ecmanage

Monthly ECManange Report: De Fashion Fabriek (L)

This month, no additional comments.



Active users and orders processed in your active client environments

Year	Month	Orders	Users
2019	4	13	162
2019	3	8	160
2019	2	12	165
2019	1	16	150
2018	12	10	150
2018	11	10	150
2018	10	7	149
2018	9	15	149
2018	8	7	139
2018	7	8	112
2018	6	9	141
2018	5	11	138
2018	4	8	138
2018	3	25	138

Users and orders per active client environment

Environment	Orders	Users
Leverancier	0	4
Angel Air	9	14
Bedrijfs Code	0	0
DC Apeldoorn	0	5
DENNISTEST107	0	0

Dennis Test 01	0	1
Ecmanage Demo omgeving	0	6
Ecmanage Test	0	0
Gemeente Den Haag	0	8
KPN	0	8
LOTOS Demo	0	5
Linero	0	11
Maatschappij voor Intercommunaal Vervoer te Brussel	0	2
Martjke Test	3	13
Mario Test	1	11
Medic Event Support	0	35
Nederlandse Spoorwegen	0	20
OWASP voor pentest	0	7
Politie	0	6
Politie	0	0
Spaans	0	0
Test	0	0
Test 00	0	0
Test 2 BV	0	0
Test BV	0	1
Test Martjke 2	0	0
Test Spaans	0	0
Thurbo	0	5
Trendworkwear	0	0
Wessa Code	0	0
britstest01	0	0
personImporttest	0	0
spanjetest01	0	0
test 2	0	0
test3	0	0

9.5 Voorbeeld van een IRIS Rapportage

softwaremakers

OIS Intern **26-06-2019 (Week: 26)**

Deze week zijn er geen opmerkingen.

Projecten

Aantal: 1

Nummer	Meldingsdatum	Titel	Behandelaren	Status
IRS10721676	25-01-2015	Ontwikkeling International Care and Research System (ICARES)	Mike van Engelen Mike van Engelen	In behandeling

Wijzigingsverzoeken

Geen Wijzigingsverzoeken bekend.

Openstaande Meldingen

Geen Openstaande Meldingen bekend.

Binnengekomen Meldingen

Geen Binnengekomen Meldingen bekend.

Opgeloste Meldingen

Geen Opgeloste Meldingen bekend.

SLA Taken

Aantal: 2

Nummer	Titel	Werkbak
IRS10403831	SLA Indexatie voor CACF	Peter Traa Peter Traa
IRS10375418	OIS R4 Referentie Implementatie patches - Merge 36	Remco Joosse Remco Joosse

OIS softwaremakers // Ampèrelaan 4 2289 CD Rijswijk // +31 (0)70 – 319 26 24 // www.ois.nl Blad 1 van 1

10 Leermomenten

De teamleader had het er steeds over dat ik even moest gaan zitten met andere programmeurs die een zelfde soort implementatie hadden gemaakt als waar ik mee bezig was. Ik kon dan wat ideeën uitwisselen en vuilkuilen ontwijken die zij hadden gevonden, ook was het handig voor in integratie van mij in het team. Ik sloeg dit meestal af omdat ik het idee had dat mijn project best wel los stond van die van hun, en dat ik makkelijk oplossingen kon vinden op het internet. Toen Remco mij nog een keer, deze keer erg dik, bij aanraadde dat ik het toch maar moest doen heb ik contact gezocht met iemand ook rapportages heeft gemaakt.

Dit gesprek was uiteindelijk best wel leerzaam! Hij gaf mij trots een demo van zijn werk, en gaf me veel tips. Ik was niet van plan zijn precieze tooling over te nemen, maar het was interessant te zien hoe hij het had aan gemaakt, en hoe hij de templates van de rapportages modulair gericht kon krijgen. Hij had, net als ik van plan was, een website gemaakt waar je verschillende instellingen kon aangeven. Hier stonden nog meer functionaliteiten op dan dat ik had bedacht, die ook goed bij mijn ontwerp konden. Ook hadden we het even over zijn hobby's, wat leuk was omdat we wat dingen gemeen hadden. Ik was best wel blij dat ik uiteindelijk met mijn college had gepraat. Ik voelde me een beetje als een tiener die eindelijk een keer naar zijn ouders (in deze vergelijking dan mij teamleider) luistert en blij was met de keuze.

Later zat ik in hele dag met een bug in mijn code. Stackoverflow hielp me niet verder en de uitlezing van gegevens ook niet. Ik was de hele dag aan het stoeien en enorm gefocuste Uiteindelijk bleek het iets heel kleins te zijn. Ik had een variable en een ander bestand aangepast, en niet goed meegenomen. Toen ik het vertelde dat ik heb aan het eind van de dag pas had dat ik hem had opgelast was mijn team leider best wel teleurgesteld. Hij benadrukte opnieuw dat er erg veel behulpzame mensen om me heen zitten die me met liefde willen helpen. Hij bedrukte opnieuw dat ik te weinig contact had met mijn collega's. Daardoor leerden zijn weinig van mij, ik weinig van hun en was het moeilijker voor mij om mee te komen in het team. Ik Begreep het wel een beetje, maar ik had een andere insteek. Mijn opdracht is erg los van andere projecten, mijn collega's hebben het erg druk, mijn bug was erg klein en moeilijk te vinden misschien en ik vond dat ik niet goed aansluit om andere dingen bij collega's.

Ik had het later nog met een vriend van mij over dit onderwerp. En vroeg om tips om te vragen hoe ik hier beter ik kon worden. Bij vertelde dat het voornamelijk om de sociale band ging, en dat ik er niet heel erg veel druk op moest leggen voor me zelf. Ook is het praten met andere mensen over bugs erg handig omdat andere mensen anders denken. Later in die week had ik een bug waarvan ik niet goed wist wat het was, en na een beetje zoeken op het internet werd ik niet heel erg veel wijzer er van. Ik besloot mijn collega aan te spreken om te vragen of hij mij wilde helpen. En dat deed hij graag. Het probleem bleek ergens te zitten waar ik niet goed over na dacht, en hij goed wist te vinden. Ik kwam er snel genoeg achter en had me die dag veel tijd gescheeld. Dit moment gaf me wat meer zelf vertrouwen om te meer hulp n te schakelen.

11 Evaluatie en reflectie

Tijdens mijn stage heb ik veel dingen geleerd. Ik heb ook veel fouten gemaakt, waar ik goed heb van kunnen leren. Hieronder zal ik mijn opdracht evalueren en reflecteren op mijn werk.

Productie evaluatie

De opdracht is zo goed als af. Het is jammer dat de helft van de opdracht, met de huidige opstelling, niet mee genomen kan worden, maar het andere deel werkt als een trein. Vooral in de laatste week heb ik veel kunnen testen de problemen kunnen uitroeien. Hier en daar zijn nog wat features die mooi geweest waren (IRIS mailen iedere vier weken en specifieke rapportage generatie), maar hier had ik in een eerder stadium mee moeten beginnen om ze erin te verwerken.
Over het algemeen ben ik wel trots op mijn opdracht. Het werkt goed, en heeft veel ingewikkelde processen en mooi met elkaar samenwerken.

Procesevaluatie

De producten zijn grotendeels zelfstand tot stand gekomen. Een begaande fout in het proces was de infrequente gesprekken met de opdrachtgever. Dit is groten deels mijn schuld. Ik was erg nerveus over mijn code en kon moeilijk een moment kiezen waar ik het goed genoeg vond om te laten zien. Achteraf gezien maakte dat niet zoveel uit, en hadden gesprekken mij kunnen leiden naar een betere focus en een ronden eindproduct. Wel heb ik me goed kunnen houden aan mijn schema en heb ik grote deadlines en milestones goed gehaald.

De eerste twee maanden waren vooronderzoek, terwijl ik maar een maand daarvoor had gepland aan het begin. Achteraf gezien was die tweede maand wel handig, want daar heb ik veel dingen uitgeprobeerd en ene betere keuze kunnen maken over mijn pakket samenstelling. Als ik direct was begonnen was ik waarschijnlijk gebleven bij ArangoDB en had ik nooit MongoDB en Spring Boot aangeraakt. Uiteindelijk kon ik redelijk want ontwerpen, en modules, van mijn prototype overnemen naar mijn eindproduct, want mij tijd scheelde later.

De RUP methode kwam erg goed van pas bij mijn ontwikkeling. De duidelijk afgebakende processen maakte, en de volgorde van de stappen, zorgde er voor dat ik een goed overzicht had over hoe ik wilde door lopen. Het grootste blok, implementatie, ging redelijk van zelf. Via Trello kan ik goed overzicht houden over de dingen die ik nog moest doen, en kon ik notities voor me zelf achterlaten voor dingen die ik nog moest doen. Af en toe kon ik overleggen met Collega's of ik nog wel op het rechte pad zat, of dat ze misschien een beter oplossing hadden. Dat was handig want het heeft me een paar keer veel tijd gescheeld.

Ik ben vooral tevreden over mijn groei als ontwikkelaar. Ik heb veel geleerd over verschillende ontwerp theorieën en heb meer visie gekregen op een grote schaal. Op de HBO heb ik ook veel projecten gedaan, maar die waren erg resultaat gefocust. Deze opdracht was vooral gebruik gefocust. Ik moest echt kijken waar het voorwas, en hoe ik het goed kon laten draaien/uitbreiden. In het begin vond ik dat heel erg moeilijk, maar langzamerhand werd ik er beter. Nu kan ik ook goed inzien waarom sommige ontwerpen foutgevoellig, of onrobuust zijn.

Ik werd erg nerveus van grote bugs. Het is regelmatig voorgekomen dat ik een lange tijd vast zat op iets kleins, en dat het me niet lukte op uit te vinden wat het was. Ik ging dan snel het ergste denken wat niet hielp in het latere proces. Ik was bang dat ik mijn hele opdracht niet ging afkrijgen en dingen die ik moest omzetten. Toen ik eindelijke de bug had opgelost bleek dat ik nog prima op schema lag en nog alles kon afkrijgen.

Voor een volgend project denk ik dat ik eerst beter naar de Scope moet kijken van het project. Ik was iets te enthousiast deze keer en wilde snel beginnen. Door eerst meer te praten met de opdrachtgevers en stakeholders had ik een ronder beeld kunnen krijgen van wat echt het doel was. Ook moet ik volgende keer vaker praten met mijn opdracht gever over de gang van zaken. Op deze manier kan ik bij opdracht beter sturen naar een eindproducten volgens zijn/haar wensen en voorkom ik overbodig werk.

Beroepstaken

- A-**1 Analyseren probleemdomein & opstellen probleemstelling**
    - Dit was een groot deel van mijn stage. Het was moeilijk en ik heb mijn best gedaan.
- **A-2 Informatie vergaren, analyseren, beoordelen & verwerken**
    - Ik heb twee maanden onderzoek gedaan naar pakketkeuze en ontwerp. Hier heb ik veel geleerd van de opties en het maken van geode keuzen.
- **A-3 Vergaren en analyseren van requirements**
    - Ik heb in de eerste weken veel geanalyseerd en requirements opgedaan. Het ging goed, maar had misschien beter kunnen kijken.
- B-4 Gemotiveerd selecteren van ICT gerelateerde oplossingen
    - Ik heb goed nagedacht over heel veel van mijn keuzes, wat vaak voor goede besluiten heeft gekozen.
- C-6 Ontwerpen software
    - Ik heb eerst goed nagedacht voordat ik mijn software implementeerde.
- D-14 Realiseren van software
    - Na het prototypen was het duidelijk wat ik wilde hebben, en hoe ik het moest maken.
- D-15 Testen
    - De laatste twee weken heb ik veel van mijn software getest en veel errorhandling geïmplementeerd. Voor echt test modules was geen tijd meer.
- D-16 Realiseren & gebruiken database
    - Ik heb een compleet database systeem ontworpen en aangesloten op een andere. Deze taak ging zeer goed.
- E-19 Beheersbaar laten verlopen van ontwikkelprocessen
    - Met hulp van RUP heb ik mijn stage goed doorlopen.
- **G-a Effectief (Internationaal)Communiceren**
    - Ik moest veel communiceren met mijn nieuwe collega's. Mijn opdracht was vooral zelfstandig, maar heb wel vaak kunnen overleggen met collega's. Wel was ik nog nerveus voor contact met mijn opdrachtgever omdat ik het (misschien te) goed wilde doen.
- **G-c Kritisch, onderzoekend & methodisch werken**
    - Ik heb na het prototypen een grote switch gemaakt en pakketkeuze om die eerder gemaakte keuze niet voldeed aan mijn wensen.
- G-e Innovatief en creatief werken
    - De opdracht was nog redelijk vrij, dit gaf mij veel ruimte voor creatieve keuzes en oplossing. MongoDB was een van deze creatieve keuzes.
- **G-f Leren leren: voorbereiden op volgende studiefase en beroep**
    - Ik heb heel veel geleerd over ontwikkelen en ontwikkelmethodes. Ik heb het gevoel dat ik in deze taak de meeste voortang heb geboekt.

12 Nawoord

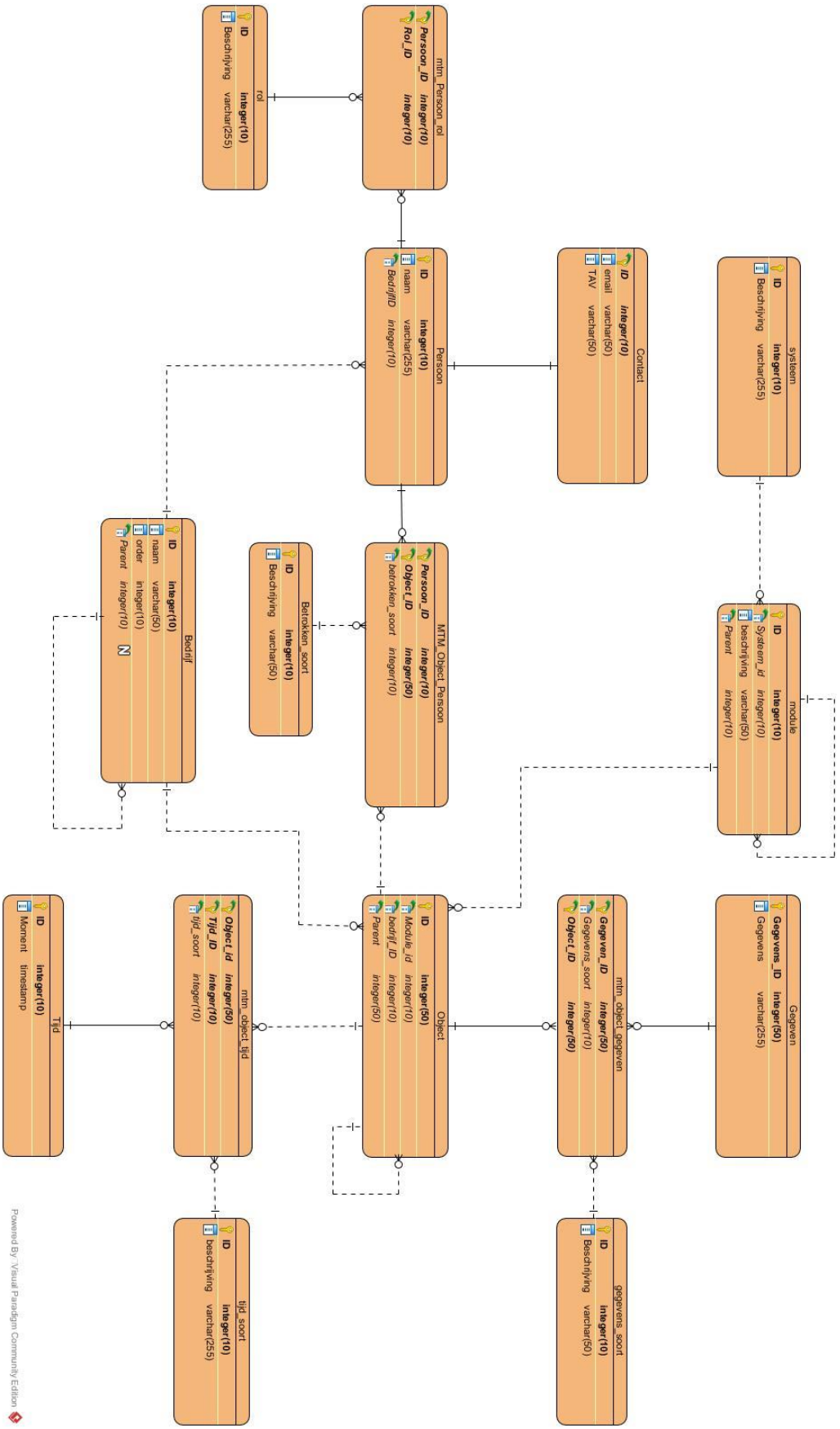
13 Literatuur

Lekberg, D., & Danielsson, P. (2013). *Designing and Implementing Generic Database Systems Based on the entity-attribute-value Model.* Karlstad: Karlstads University, Faculty of Health, Science and Technology.

West, M., & Julian, F. (1996). *Developing High Quality Data Models.* The European Process Industries STEP Technical Liaison Executive (EPISTLE).

14 Bijlagen

14.1 A: ERM Ontwerp



14.2 B: Proof-of-Concept Dataset

NB: Alle beschreven gegevens zijn placeholders en zijn geen klanten van OIS.

IRIS Meldingen:

IKEA – Rapportage 101

Soort	IRIS_Nummer	Meldingsdatum	Titel	Behandelaren	Status	Module	#
Binnen	IRS88954320	2014-04-20	Kijkfunctionaliteit binnen ECManage	Michiel Maas	-	-	5

IKEA – Rapportage 102

Soort	IRIS_Nummer	Meldings datum	Titel	Behandelaren	Status	Module	#
Project	IRS12345678	2014-05-07	Samenwerkings-Project	Dennis Coert	In behandeling	-	1
Wijziging	IRS12365426	2014-05-02	Routecodes	Leo Vonk	Onderweg	-	2
Wijziging	IRS15398763	2016-09-07	Support voor database	Marijke Kniest	Preproductie	-	3
Opgelost	IRS88953012	2019-02-19	Mail versturen na budgetopdracht	Nicole Kramer	-	Koppeling	6

OIS – Rapportage 102

Soort	IRIS_Nummer	Meldings datum	Titel	Behandelaren	Status	Module	#
Open	IRS11123654	2017-09-11	Bestellen op verkeerde adres	Marijke Kniest, Dennis Coert	-	Bezorgmodule	4
Taak	IRS99651435	-	Koffie Halen	Michiel Maas	-	-	7

IRIS Rapportages:

Rapportage Ronde	Bedrijf	Meldingen	Contact Persoon
101	IKEA	5	Bork Malle
102	IKEA	1, 2, 3, 6	Bork Malle, Knacke Bork
102	OIS	4, 7	Evert van Es

Uitzonderingen:

- IRIS_Nummer IRS88954320 – Vorige ronden, hoort niet meer mee te komen
- IRIS_Nummer IRS11123654 – Twee Behandelaren
- Twee wijzigingsverzoeken in Rapportage 102 van de IKEA
- Geen nieuwe meldingen
- Gegevens zijn aan de hand van de tabellen die gemaakt worden met de Macro's

ECManage Gegevens:

Bedrijven:

- Simple Workwear
  - Praxis – Actief over de hele periode
  - Albert Heijn – Actief tussen April 2018 en Juli 2018
- NS
  - Pro-rail – Actief sinds November 2018

Verkoop Gegevens Simple Workwear:

	Praxis		Albert Heijn		Simple Workwear	
	Dragers	Orders	Dragers	Orders	Dragers	Orders
2018-01	3	9	-	-	3	9
2018-02	4	10	-	-	4	10
2018-03	5	11	-	-	5	11
2018-04	27	3	200	10	227	13
2018-05	30	5	300	26	330	31
2018-06	35	25	333	4	368	29
2018-07	60	18	357	15	417	33
2018-08	90	32	-	-	90	32
2018-09	60	40	-	-	60	40
2018-10	110	30	-	-	110	30
2018-11	120	45	-	-	120	45
2018-12	200	90	-	-	200	90
2019-01	250	100	-	-	250	100
2019-02	350	50	-	-	350	50
2019-03	351	20	-	-	351	20

Verkoop Gegevens NS:

	Pro-rail		NS	
	Dragers	Orders	Dragers	Orders
2018-11	25	4	25	4
2018-12	30	8	30	8
2019-01	75	10	75	10
2019-02	100	26	100	26
2019-03	110	30	110	30

Opmerkingen:

- Er zijn twee leveranciers en drie omgevingen
- Albert Heijn was een tijdje actief, en moet geen maand overzicht krijgen in het resultaat
- Voor NS moeten we alleen deze maanden geven, en die leeg tot afgelopen jaar

Contact Personen:

Persoon	TAV	EMAIL	Bedrijf	Moet mail krijgen van
Bork Malle	“Hej da Bork,”	borkmalle@IKEA.nl	IKEA	IRIS Rapportage 15 en 102
Knacke Bork	“Hej da Knacke”	knackebork@IKEA.nl	IKEA	IRIS Rapportage 102
Evert van Es	“Beste meneer van Es,”	Evert.van.es@ois.nl	OIS	IRIS Rapportage 102 en ECManage Sales Totals
Treintje Oosterhuis	“Beste mevrouw Oosterhuis, ”	treintje@ns.nl	NS	ECManage Overzicht voor NS
Guus Meeuwis	“Hallo meneer Meeuwis”	Guus_m@ns.nl	NS	-
Freek Vonk	“Gegroet meneer Vonk, ”	f.vonk@sw.nl	Simple Workwear	ECManage overzicht voor Simple Workwear

Opmerkingen:

- Bork Malle en Knacke Bork krijgen IRIS Rapportage 102
- Guus Meeuwis krijgt geen berichten
- Evert van Es krijgt overzicht van ECManage en IRIS

Overzichten generen:
Met deze gegevens moeten een aantal overzichten gemaakt worden. Deze overzichten hebben een aantal eisen en een bepaalde vorm:

- IRIS Rapportage:
    - Alle meldingen uit het systeem van afgelopen week, verdeeld per soort
    - Alle correcte gegevens van de meldingen beschreven (volgens bestaande format)
    - Verstuurd bericht naar de goede mensen
- EC Manage Monthly
    - Overzicht per leverancier van de verkoop gegevens (Dragers en Orders) van afgelopen 12 maanden bij elkaar
    - Gegevens van openstaande omgevingen
    - Verstuur bericht naar de goede mensen
- EC Manage Sales Totals
    - Overzicht van de verkoop gegevens (Dragers en Orders) van alle leveranciers bij elkaar van de afgelopen 12 maanden bij elkaar
    - Gegevens van openstaande omgevingen per leverancier beschreven.
    - Verstuur bericht naar de goede mensen

Mogelijkheden:

- Aanpassen van de mensen die de berichten moeten krijgen
- Toevoegen van een nieuwe omgeving bij een goede leverancier

14.3 C: SQL Query voor IRIS Rapportages

```
SELECT object.ID AS HM, (
SELECT vw_object_gegeven.Gegevens
FROM object
JOIN vw_object_gegeven ON object.ID = vw_object_gegeven.ObjectID
WHERE vw_object_gegeven.GegevenSoort = 0 && vw_object_gegeven.ObjectID = HM
                ) AS 'Soort', (
SELECT vw_object_gegeven.Gegevens
FROM object
JOIN vw_object_gegeven ON object.ID = vw_object_gegeven.ObjectID
WHERE vw_object_gegeven.GegevenSoort = 1 && vw_object_gegeven.ObjectID = HM
                ) AS 'IRS_Code', (
SELECT vw_object_gegeven.Gegevens
FROM object
JOIN vw_object_gegeven ON object.ID = vw_object_gegeven.ObjectID
WHERE vw_object_gegeven.GegevenSoort = 2 && vw_object_gegeven.ObjectID = HM
                ) AS 'Titel', (
SELECT vw_object_gegeven.Gegevens
FROM object
JOIN vw_object_gegeven ON object.ID = vw_object_gegeven.ObjectID
WHERE vw_object_gegeven.GegevenSoort = 3 && vw_object_gegeven.ObjectID = HM
                ) AS 'Module', (
SELECT vw_object_gegeven.Gegevens
FROM object
JOIN vw_object_gegeven ON object.ID = vw_object_gegeven.ObjectID
WHERE vw_object_gegeven.GegevenSoort = 4 && vw_object_gegeven.ObjectID = HM
                ) AS 'Status', (
SELECT vw_object_naar_meldingsdatum.Moment
FROM vw_object_naar_meldingsdatum
WHERE vw_object_naar_meldingsdatum.ID = HM
                ) AS 'Datum',   (
SELECT persoon.Naam
FROM object
JOIN mtm_object_persoon ON  object.ID   = mtm_object_persoon.Object_ID
JOIN persoon ON     persoon.ID  = mtm_object_persoon.Persoon_ID
WHERE object.ID = HM && mtm_object_persoon.Soort_id = 1
            ) AS 'Programmeur', (
SELECT persoon.Naam
FROM object
JOIN mtm_object_persoon ON  object.ID   = mtm_object_persoon.Object_ID
JOIN persoon ON     persoon.ID  = mtm_object_persoon.Persoon_ID
WHERE object.ID = HM && mtm_object_persoon.Soort_id = 3
            ) AS 'Contactpersoon'
FROM object
WHERE object.Parent IN (
    SELECT object.ID
    FROM object
    JOIN mtm_object_gegevens ON mtm_object_gegevens.Object_ID = object.ID
    JOIN gegeven ON gegeven.ID = mtm_object_gegevens.Gegeven_ID
    WHERE object.ModuleID = 2 && mtm_object_gegevens.Gegeven_Soort && gegeven.Gegevens
    IN (
    SELECT *
    FROM vw_huidge_rapportage_ronde))
```

14.4 D: Slides van Database Keuze Presentatie

RDBM vs noSQL

presentatie voor stage OIS Michiel Maas

Introductie

- Belangrijke beslissing
- Opslag voor data
- Generieke database
- Geschrokken van Datamodel, op zoek naar alternatief
- Afwegen soorten
  - RDBM (MariaDB)
  - noSQL (ArangoDB)
  - Key-Value (EAV)

RBDM vs noSQL

- RBDM
  - mariaDB
  - Klassiek SQL
  - Sterk genormaliseerd door generieke eis
  - veel joins als gevolg

- noSQL
  - ArangoDB
  - Multi Model (Document, Graph en Key-Value)
  - Enorm gedenormaliseerd
  - Veel redundantie
  - Eigen query language (AQL)
  - noSQL, maar dan met joins

EAV / Key-Value

- Super generiek
- Heel veel werk in- en uitvoer
- Niet meerdere Foreign Keys voor datatype

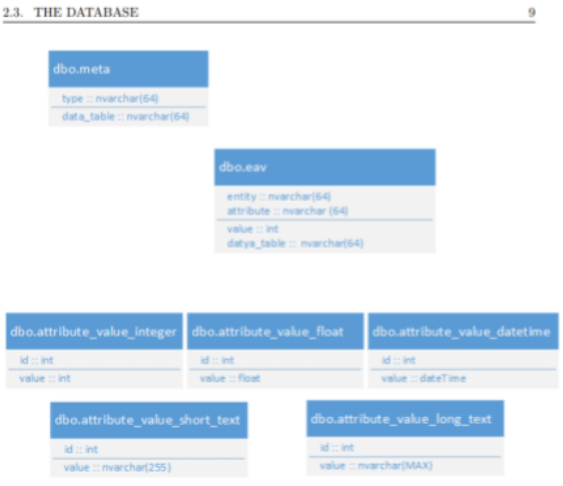


Figure 2.3: UML diagram of our database design

Eisen aan proof-of-concept

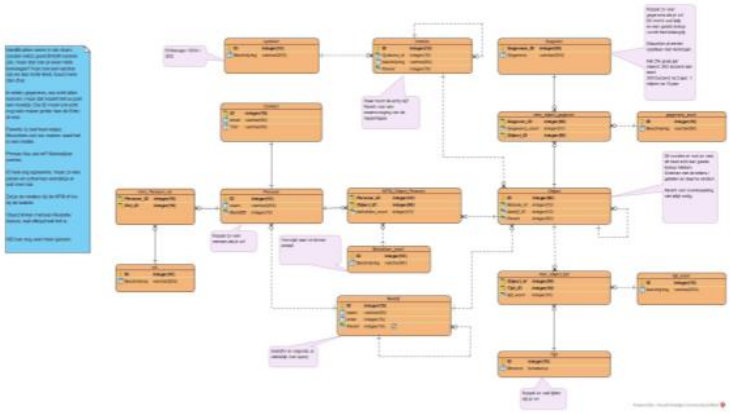
- Genereren van overzichten
  - IRIS Rapportage
  - ECManage Sales Maandelijks
  - ECManage Sales Total
- Gebruik dezelfde dataset
- Test Updating
- Zo min mogelijk workaround / hardcoded
- Schrijf queries voor groei
- Probeer verschillende opties in data ophalen

Dataset en uitzonderingen

- IRIS Rapportage
  - Twee bedrijven
  - Een melding van vorige ronde
  - Meerdere behandelaren en contactpersonen
- ECManage Monthly
  - Bedrijf dat niet meer mee doet
  - Bedrijf dat korter mee speelt
- ECManage Sales Totals
  - Omgeving die niet meer mee doet

RDBM - Datamodel

- Generiek, maar met een beetje structuur
- Object entity
- many-to-many met verschillende attributen
- Relaties zijn beschreven in een losse tabel
- gegeven tabel word gigantisch



RDBM - Demo

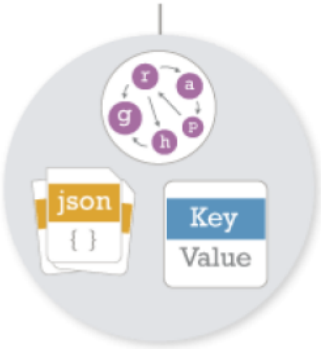
- Queries zijn lang en breed, want gegevens komen verticaal en niet horizontaal
- Gegevens zijn heel erg verspreid
- Schrijven van queries is makkelijk, bedenken is moeilijk
- Veel joins en veel views, slecht voor grote datasets
- Is snel, maar weet niet zo goed hoe dat schaalt met grotere datasets
- Invoeren is wel lastig, moet veel tables raadplegen
- Uitlezen van gegevens is niet zo elegant
- Indexing is belangrijk
- Redelijk wat hardcoded

Verbeteringen

- Vroege indicatie
- Soort beschrijving hardcoded doen, slaat join over
- Triggers & Procedures
- Materialized Views

noSQL - Datamodel

- Documents
  - JSON Bestanden met alle gegevens
- Graph
  - Verbinding tussen documenten
- Key-Value
  - Database voor snelle look-up



Ontwerp
- Losse gegevens (Sales / IRIS Melding)
- Rapport (Verzameling van de losse gegevens die erbij horen)
- Graphs tussen de gegevens en de rapportages

Gegevens kan direct, ongenormaliseerd ingevoerd worden

noSQL - Demo

- AQL duurt wel een tijdje om te leren, en werkt niet altijd mee
- Schrijven van queries is moeilijk, bedenken is makkelijk
- Is nu langzaam, maar scaled misschien beter
- Invoeren is super makkelijk
- Uitlezen is makkelijk, want je krijgt alles dat je wil hebben

Verbeteringen
- Beter queries schrijven
- Meer gebruik van Graphs en Indexes
- Meer nested/gedenormaliseerde data misschien

Uitvoer van programma

	IRIS Rapportage		ECManage Totals		ECManage Monthly	
	Empty	Cached	Empty	Cached	Empty	Cached
RDBM	579	140	206	160	240	182
noSQL	307	300	420	368	354	234

- noSQL database heeft een groot verschil in werking
- Queries zelf duren ongeveer even lang
- MariaDB heeft een cache

RBDM vs noSQL

- RBDM
  - + Snel
  - + Makkelijk
  - + Kan gemakkelijk beter
  - + Updating is veilig
  - - Schaalt misschien slecht
  - - Onelegant
  - - Invoer is ingewikkeld

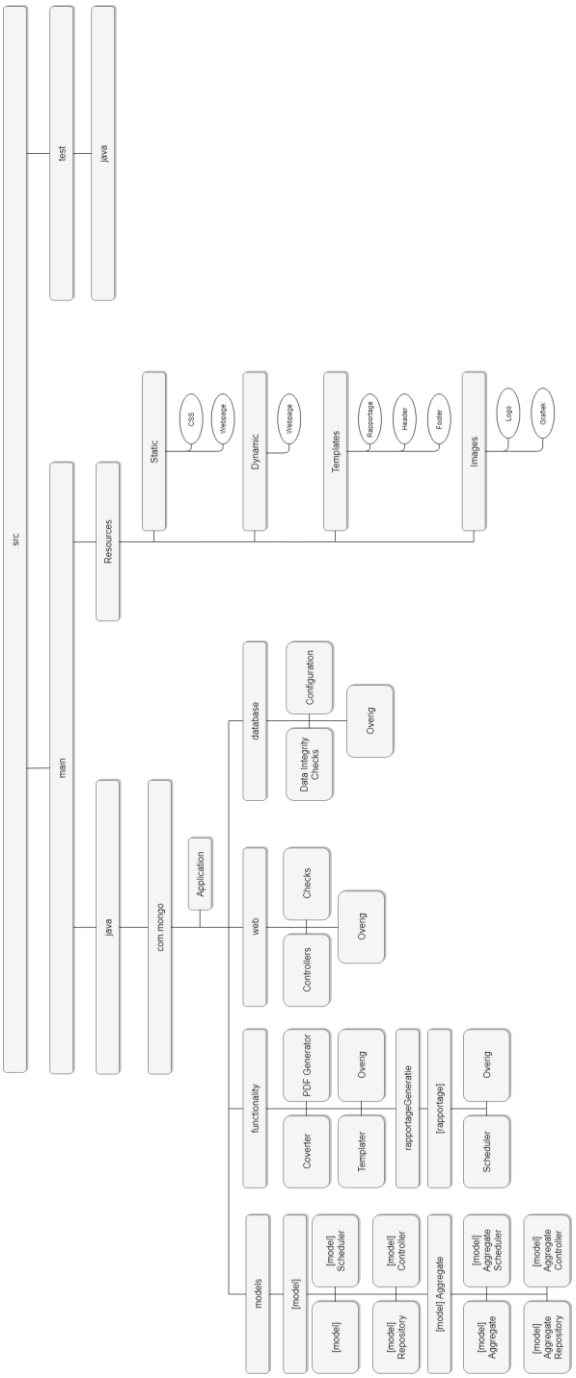
- noSQL
  - + Schaalt misschien beter
  - + Invoer is makkelijk
  - + Heeft veel functionaliteit
  - - Langzaam
  - - AQL is Lastig
  - - Verbetering is ingewikkeld
  - - Updating is onhandig door denormalisering

Volgende Project

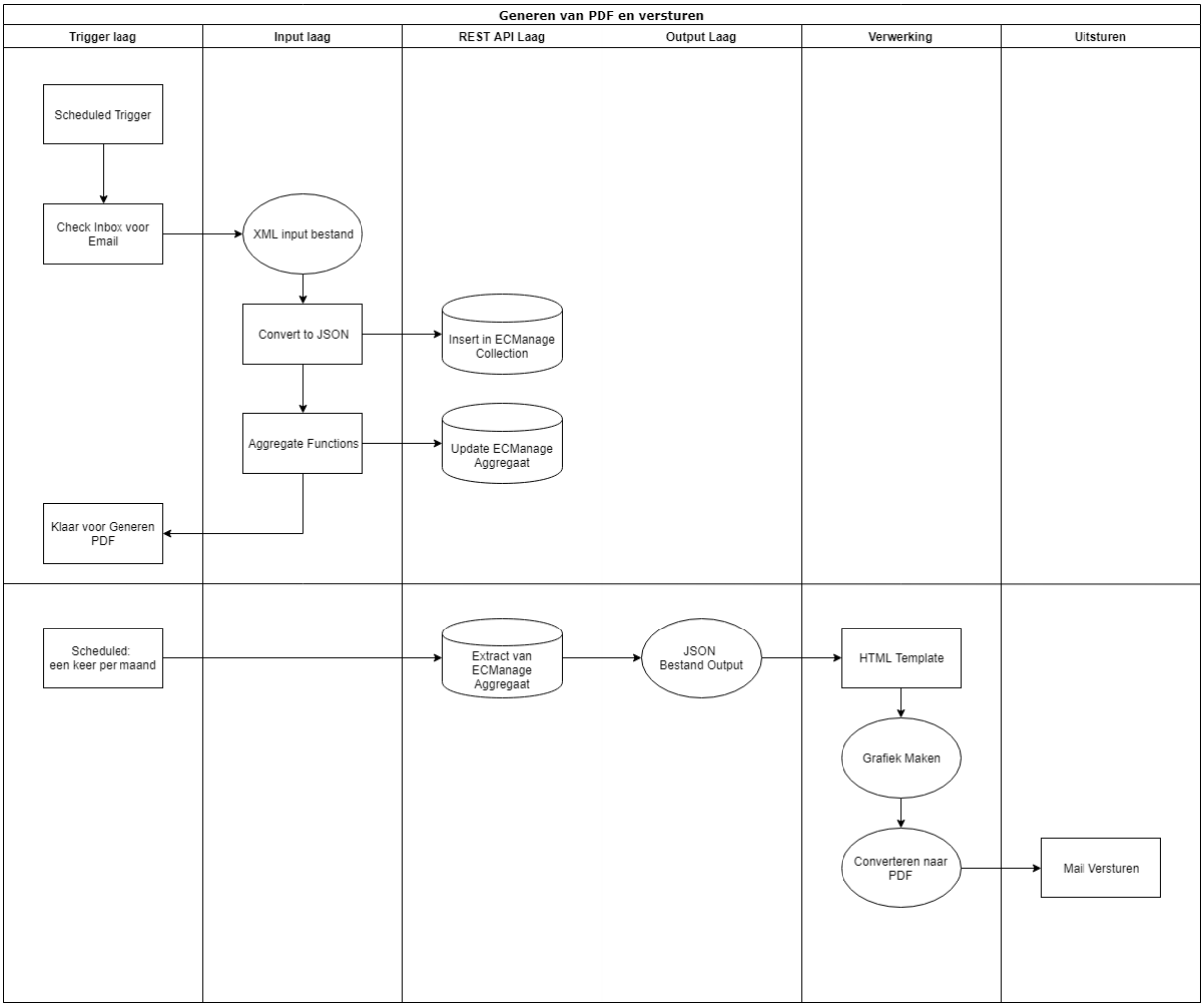
Server

Hoe houd ik de applicatie lopend en los?

14.5 E: Design Diagrammen



Figuur 1 Lay-out Code Bestanden



Figuur 2 Stappenplan generen van PDF en versturen